MANAGEMENT SCIENCE
Vol. 53, No. 8, August 2007, pp. 1315–1331
ISSN 0025-1909 | EISSN 1526-5501 | 07 | 5308 | 1315

informs®
DOI 10.1287/mnsc.1060.0687
© 2007 INFORMS

Learning from Experience in Software Development: A Multilevel Analysis

Wai Fong Boh
Nanyang Business School, Nanyang Technological University, Singapore 639798, Republic of Singapore, awfboh@ntu.edu.sg

Sandra A. Slaughter
David A. Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, sandras@andrew.cmu.edu

J. Alberto Espinosa
Kogod School of Business, American University, Washington, District of Columbia 20016-8044, alberto@american.edu

This study examines whether individuals, groups, and organizational units learn from experience in software development and whether this learning improves productivity. Although prior research has found the existence of learning curves in manufacturing and service industries, it is not clear whether learning curves also apply to knowledge work like software development. We evaluate the relative productivity impacts from accumulating specialized experience in a system, diversified experience in related and unrelated systems, and experience from working with others on modification requests (MRs) in a telecommunications firm, which uses an incremental software development methodology. Using multilevel modeling, we analyze extensive data archives covering more than 14 years of systems development work on a major telecommunications product dating from the beginning of its development process. Our findings reveal that the relative importance of the different types of experience differs across levels of analysis. Specialized experience has the greatest impact on productivity for MRs completed by individual developers, whereas diverse experience in related systems plays a larger role in improving productivity for MRs and system releases completed by groups and organizational units. Diverse experience in unrelated systems has the least influence on productivity at all three levels of analysis. Our findings support the existence of learning curves in software development and provide insights into when specialized or diverse experience may be more valuable.

Key words: software development; knowledge work; knowledge workers; organizational learning; learning curve; multilevel analysis
History: Accepted by Rajiv Banker, information systems; received July 3, 2003. This paper was with the authors $10\frac{1}{2}$ months for 2 revisions. Published online in *Articles in Advance* July 20, 2007.

1. Introduction

Although software systems are critical to almost every aspect of modern life, systems development projects are frequently behind schedule and over budget (Wastell 1999). It is striking that these problems have persisted despite automated tools, advances in programming languages and methods, and formal education and training in computer science and information systems (Brooks 1995). One may argue that such advances are not sufficient to improve the rate of successful systems development. Rather, software projects remain susceptible to failures because organizational members fail to learn from their own experience (Lyytinen and Robey 1999). Learning is crucial to the success of systems development (Wastell 1999). A learning perspective highlights the importance of considering systems development as a recurring process and encourages investment in building long-term capabilities.

In this study, we adopt the learning-curve approach to investigate the extent to which individuals, groups, and organizational units engaged in software development actually learn from their experience. Further, we consider what types of experience maximize learning rates. The organizational learning curve has been the subject of extensive study in many fields and several industries (Argote et al. 1990). The basic principle underlying the learning curve is that production experience creates a growing stock of knowledge that can be applied to improve the productivity of the organization (Argote 1999). The learning curve focuses on examining how organizations learn to work more efficiently.

Prior research has examined learning curves in manufacturing (e.g., Argote et al. 1990) and service industries (e.g., Pisano et al. 2001, Reagans et al. 2005). As industrial organizations become knowledge-based organizations, it is important to examine whether the advantages of experience evident in manufacturing tasks can be achieved in knowledge work. The learning curve assumes that as a worker performs a task repeatedly (such as in manufacturing and assembly operations, or in the making of pizzas (Darr et al. 1995)), the direct labor requirements to complete a subsequent unit of the task decline. Unlike manufacturing tasks, which are often repetitive (Argote 1999), or service work, which is frequently scripted, knowledge work entails less routinization and often requires problem solving. Compared to more unstructured forms of knowledge work like research or design, however, systems development is more routine and structured, as it makes use of methodologies and rules. Nevertheless, systems development is a knowledge-intensive activity that requires a considerable amount of abstract, technical, theoretical, and experiential knowledge (Sacks 1994). Although each systems development project is different and customized to the requirements of users (Basili and Caldiera 1995), developers can gain practice with each project and can construct intellectual schemas based on their technical and experiential understanding of software development tasks (Sacks 1994). Thus, there is some potential for experience-based learning in software development, although the extent of that learning is unknown.

Our study contributes to the software development literature by determining how learning from experience affects software development productivity for individuals, groups, and organizational units. It is important to understand the role that experience plays in software development because the largest component of software costs is for labor, i.e., the developers. Thus, insights into the productivity impacts of experience-based learning in software development have implications for how much managers should value the experience of software developers, and the extent to which managers should factor in the learning curve in project planning.

Our study also contributes to the learning-curve literature by examining learning at multiple levels of analysis. Learning from prior experience can occur at different levels of analysis within a software development organization. Although learning curves have been found to exist at the individual, group, organizational, and industry levels of analysis (Argote 1999), prior studies have not considered learning at several levels of analysis simultaneously. It is important to make this consideration because many organizational activities inherently involve multiple levels of analysis (Hofmann 1997). Further, learning processes can differ across levels (Crossan et al. 1999), and the types of experience and their impacts on productivity can also differ at different levels.

Finally, our study contributes to the organizational learning literature by examining which kinds of experience best facilitate the learning of individuals, groups, and organizational units. In particular, we investigate if developers maximize their learning by specializing in a specific system, or by diversifying their experience through working on other related and unrelated systems. Accumulating specialized experience creates reliability in experience, thus enabling organizations to exploit their experience, whereas diversifying experience creates variety in experience, thus enabling organizations to engage in exploration (Holmqvist 2004, March 1991). Exploration and exploitation are both essential processes for organizations, but they compete for scarce resources (Gupta et al. 2006). We thus explore the exploitation versus exploration trade-offs by examining whether individual experience can best be leveraged through specialization or diversification. In §2, we describe the research setting—a large systems development effort at a telecommunications firm. Then in §3, we develop hypotheses that posit learning from experience in systems development at the three levels of analysis: individual, group, and organizational unit. Sections 4 and 5 describe the empirical evaluation of our hypotheses and the analysis of data collected in the field study. In §§6 and 7 we present and discuss the results.

2. The Research Context

To examine learning at the individual, group, and organizational-unit levels within a software organization, we conducted a field study of an organization performing large-scale systems development in the telecommunications industry. We focused on a key telecommunications software telephony product developed by the organization. The product contains several million lines of code (written in the C programming language) with over half a million updates that have added new features or modified existing features. The product includes 63 major systems and 26 smaller systems. Each system was developed by a separate organizational unit that varied in size from 30 to about 300 employees. Organizational units were located at different sites and effectively functioned like subsidiaries of the company. Within each organizational unit, developers worked as individuals or in groups. Developers were sometimes rotated across several systems so that the organization could cater to fluctuating demands for resources in different systems. Hence, many developers had experience working on more than one system.

Given the size and complexity of the product, the organization used an incremental software development process, which is commonly used in many large-scale development efforts to progressively develop, evolve, and enhance systems over time (Rajlich 2006). In an incremental process, a *release* is a major product upgrade that affects one or more systems, and includes new software features and modifications to existing features. Different releases are introduced to the market at different times. To manage this incremental development process, the organization used *modification requests* (MRs), which are similar in concept to work orders. The initial development of the first release of the product was completed as a series of MRs. Subsequently, MRs were used to add new functions or modify and repair an existing function. A set of MRs representing a significant improvement to the software then makes up a new release of the software. All MRs were formally reviewed by a change committee before being approved and assigned to individuals or groups.

3. Theory and Hypotheses

In completing a release, work and learning can occur at multiple levels in the organization, thus allowing us to differentiate between three levels of analysis. At the individual level, developers complete individual development tasks. At the group level, teams of developers complete tasks that are interdependent. At the organizational-unit level, tasks performed by individual developers and groups of developers in the organizational unit complete the requirements for a release. In this section we develop hypotheses about how different types of experience influence productivity at the individual, group, and organizational-unit levels of analysis in systems development.

3.1. Learning from Individual Experience

Knowledge and skills are not only important inputs for software development work, but are also important outputs, because they are continuously enhanced with experience. Designing, coding, and testing of software are complex and difficult cognitive tasks that build on practical experience. As individuals accumulate experience by completing more units of work, they become more proficient in software development. We identify three types of cumulative experience acquired by individuals that are expected to affect productivity at the individual, group, and organizational levels of analysis: (1) Experience in a specific system; (2) experience in related systems; and (3) experience in unrelated systems.

Experience in a Specific System. In systems development, specialization refers to the accumulation of experience about a specific system. Specialized experience with a particular system increases developers' familiarity with the architectural domain of the system (Banker et al. 1998), including the structure of the components, files, and code within the system, and the linkages between them (Robillard 1999). In incremental systems development where new code has to be integrated with existing code, understanding the given component and its interrelations with other parts of the system is particularly important (Rajlich 2006). Developers spend a significant amount of time comprehending the source code of the system being changed or enhanced. Developers' existing knowledge of the structures of the files and components in specific systems will thus afford them with much greater efficiency in comprehending those systems and in updating, enhancing, and adding new components to the systems (Ramanujan et al. 2000).

Experience in Related and Unrelated Systems. Prior research in psychology has found that individuals can spontaneously develop a new understanding of a problem because they transfer knowledge from one domain to another: What was well understood in one problem domain suddenly provides an analogous solution to a new problem domain (Schilling et al. 2003). This transfer of knowledge may occur across domains that appear to have little in common (Simonton 1999). In systems development, developers can increase the diversity of their experience by working on a variety of related and unrelated systems. We expect developers to be able to transfer and apply the experience gained from developing one system to the development of other systems.

A key question affecting learning from experience is whether more leverage can be gained by specializing, where individuals focus on one kind of task, or by diversifying, where individuals switch between multiple kinds of tasks. Specialization allows individuals to complete more repetitions of a task within a given time, and to gain an in-depth understanding of the problem domain. An individual switching between multiple kinds of tasks—even if the tasks are related—might become distracted from learning concepts that apply only to the core task (Schilling et al. 2003). In contrast, diverse experience allows individuals to gain breadth of knowledge, which can increase their absorptive capacity, as varied prior learning improves individuals' ability to evaluate and utilize outside knowledge (Cohen and Levinthal 1990). Although developers can potentially learn from both specialized experience in one system and diverse experience in related and unrelated systems, they have limited time. Hence, there are trade-offs involved in deciding whether they should accumulate more experience in specific systems, or whether they should be working on a variety of related and unrelated systems (Gupta et al. 2006). Despite the importance of understanding the implications of this trade-off, the impact of specialization on organizational learning has received scant theoretical or empirical attention (except for Schilling et al. 2003). In the following sections, we develop hypotheses to understand the relative importance of each type of individual cumulative experience in facilitating productivity, and explore whether the relative importance differs at the individual, group, and organizational-unit levels of analysis.

3.1.1. Learning at the Individual Level of Analysis. Research in psychology has found that learning across tasks can contain two aspects: the knowledge content transfer (e.g., knowledge about the code and architecture of a system) and the learning process transfer ("learning to learn") (Schilling et al. 2003, p. 42). *Learning to learn* is the process by which individuals improve their ability to learn over time because they transfer their previous learning about how to apply particular kinds of information to a new problem (Ellis 1965, Schilling et al. 2003). As developers accumulate experience within a system, they can transfer both knowledge content and learning processes. We expect developers to gain the most leverage from specializing in one system as increasing familiarity with the code and architecture in the system has a direct impact on their ability to understand, enhance, and modify the code (knowledge-content transfer). Although diverse experience in related and unrelated systems can improve the breadth of developers' knowledge, the effect is expected to be less significant, as diverse experience in other systems may not directly apply to the current system and would require effort to adapt to the current system. Thus:

Hypothesis 1A (H1A). *Developers' current productivity in completing individual work is more positively affected by their prior experience in the* same system *than by their prior experience in* related or unrelated systems.

Studies of individual learning in psychology have demonstrated that related task variation (varying task content or context) may enhance an individual's ability to learn by facilitating the development of abstract principles that can be applied to different, but related, problems (Schilling et al. 2003, Schmidt 1975). For example, engineers often need to explore a problem in several different settings (such as in both the plant and the laboratory) before they are able to understand and resolve it (Schilling et al. 2003, Tyre and von Hippel 1997). Similarly, in software development, experience in related systems can allow developers to abstract an overall cognitive structure and schema of the dependencies and linkages between related systems. Individuals' understanding of the code, structure, and impact of changing the code in the current system can thus be reinforced and improved. Developers also learn to learn by improving their understanding of general approaches to problem solving, general principles about programming, and the effective usage of organizational software processes and tools. Encountering similar problems in related systems may allow developers to better abstract the general principles about programming and problem solving. Prior work on related systems may also help developers to better anticipate and avoid problems in making changes to the current system.

When developers gain experience working on unrelated systems the extent of knowledge-content transfer is small, and the extent of learning-process transfer is also expected to be limited, as the systems may be so different that it becomes difficult for developers to see the parallels with the current system.

Hypothesis 1B (H1B). *Developers' current productivity in completing individual work is more positively affected by their prior experience in* related systems *than by their prior experience in* unrelated systems.

3.1.2. Learning at the Group Level of Analysis. One key difference between working individually and working in a group is the need for sharing and integrating knowledge across group members (Brooks 1995, Crossan et al. 1999). When software developers work in groups, we expect individual specialized experience to positively influence group productivity, just as it would influence individuals' productivity when developers work alone. However, due to the need for coordinating, sharing, and integrating knowledge in a group (Crossan et al. 1999), we expect individual diverse experience to have a more significant positive impact on group productivity than specialized experience. This is because individual diverse experience is likely to improve knowledge sharing and integration within the group.

There has been extensive research examining the effect of group-member diversity on information sharing and group performance (Mannix and Neale 2005). Few researchers, however, have conceptualized diversity as the extent to which individuals in groups are narrow specialists with experience in a limited number of areas, or broad generalists with experience in several areas (Bunderson and Sutcliffe 2002). Bunderson and Sutcliffe (2002) was one exception, and they found that the functional diversity of individuals in teams had a significantly positive effect on information sharing and team performance. Building on Bunderson and Sutcliffe's (2002) research, our study thus explores the relative impact of individual specialized and diverse experiences on group productivity in the software development context.

In a software development group, when individuals have experience in a diverse set of systems, they are more likely to have a common understanding of the overall architecture of the product, and linkages between systems. Individuals are also exposed to coding and development practices used in different systems. This enables them to understand and appreciate different development practices and coding styles adopted by other developers, and reduce their biases as to whether there was one right way of doing things. Hence, diversity of individual experiences improves the willingness of group members to share information and the ability of other group members to recognize the relevance and importance of information shared by others in the group. This facilitates information sharing and coordination in the group (Bunderson and Sutcliffe 2002, Rulke and Galaskiewicz 2000). In addition, groups whose members have diverse experience are better at integrating new and diverse information to arrive at a more effective solution for their problem on hand (Brown and Eisenhardt 1995). The more individuals store objects, patterns, and concepts in their memory, the more they are able to acquire new information and use information in new settings (Bunderson and Sutcliffe 2002). In software development groups, diversity of group members' individual experience will help the groups better reconcile and recombine differences in opinion, and generate the design, code, or debugging solution for new or existing programs more efficiently.

Hypothesis 2 (H2). *A software group's current productivity in completing group work on a system is more positively affected by the average level of group members' prior experience from work on* related and unrelated systems *than from work on the* same system.

3.1.3. Learning at the Organizational-Unit Level of Analysis. At the organizational-unit level of analysis, we consider the productivity of the unit responsible for completing system releases. A system release can only be completed with the combined efforts of all individuals and groups in the unit working on the system; hence, the completion of system releases reflects the ability of the entire organizational unit to work efficiently. To complete a system release, the experience accumulated by individuals and groups involved in the system release must be integrated. Managers have to determine whether they gain more leverage from allocating developers in the unit to the same system to accumulate specialized experience, or from allocating developers to work on a variety of systems to increase their diversity of experience. In making these decisions, managers have to balance between the need for exploitation versus exploration.

Allocating developers to gain specialized experience within a particular system creates reliability in experience, allowing organizations to exploit the experience and increase individual productivity. However, it can also lead to a competency trap (Levitt and March 1988). As noted by Garud and Kumaraswamy (2005, p. 11), "in the very act of refining existing knowledge..., employees may forgo opportunities to renew and expand their knowledge tool kit." Gaining diverse experience, on the other hand, might enhance the future learning rate of the organizational unit (Schilling et al. 2003), as individuals with greater breadth of knowledge are more likely to recognize the value of new information and to assimilate it. Fichman and Kemerer (1997), for example, found that software organizations with more diverse technical knowledge have greater absorptive capacity and can assimilate new process innovations more readily. Having a greater breadth of knowledge will enable individuals to acquire and use knowledge more effectively, as prior related knowledge facilitates the assimilation of new knowledge (Matusik and Heeley 2005). This prevents developers from becoming too narrow minded, allows them to be more flexible in their approaches to software development, and facilitates their adaptation to the requirements and workings of different groups. Having a pool of developers with a greater breadth of knowledge therefore increases the flexibility of the organization to assign developers to work on different systems based on the differing needs of the organizational units. This allows each organizational unit to utilize its resources more effectively to complete system releases more efficiently.

Hypothesis 3 (H3). *An organizational unit's current productivity in completing a system release is more positively affected by the average experience of the unit's developers from work on prior releases of* related or unrelated systems *than from work on prior releases of the* same system.

3.2. Learning from Working with Others

Groups are often used in large-scale systems development because of the size and complexity of the tasks. Within a group formed to complete an MR, individual developers work together in the task. As requests for new MRs emerge, different configurations of development groups are assigned to the MRs, due to different task-expertise requirements. Each development group is thus a temporary group (Meyerson et al. 1996), consisting of a set of developers working jointly on a task over a limited time period. The need to work with other developers presents both opportunities and challenges to learning at the individual, group, and organizational-unit levels of analysis.

3.2.1. Learning at Individual Level of Analysis. Individuals can benefit from knowledge accumulated by others (Reagans et al. 2005). This can occur through processes of knowledge transfer as developers work with others. Working with others gives individuals the opportunity to learn through observations and discussions about the task. In particular, the opportunity to work with many different developers will expose individual developers to diverse practices and insights, enabling them to benefit from others' experience. However, not all developers will work with many others. One of the main principles of human communication, called selective exposure, is the strong tendency for individuals to communicate with others who are most like themselves or who are most likely to agree with them. Individuals who interact selectively with a few similar others avoid messages and information that might conflict with their established practices and dispositions (Katz 1982). In contrast, individuals who work with many different developers have more opportunities to learn from others and share knowledge more extensively, thereby improving their productivity.

Hypothesis 4A (H4A). *Developers' current productivity in completing individual work on a system is positively related to the number of different developers they have worked with in prior MRs.*

3.2.2. Learning at the Group Level of Analysis. Effective coordination is important when developers work in teams. The need for coordination is exacerbated in large-scale software development, where dependencies abound among individuals (Kraut and Streeter 1995). Expertise coordination and group cognition have been found to have significant impact on group productivity (Espinosa 2002, Faraj and Sproull 2000). As collaborators interact over time, they develop common knowledge about tasks, goals, and strategies that facilitate their work, which helps them to manage tasks and member dependencies more effectively. As some or all of the developers within a group work together in prior MRs, their interactions in the past help them to develop familiarity with one another, learn about who is good at doing what (Liang et al. 1995, Wegner 1986), and learn to coordinate with one another. For example, software groups can organize their work in such a way that the structure of the work accommodates the strengths and weaknesses of the members. This requires knowledge about who is good at doing what in the team—built through individuals' prior experience in working with one another (Liang et al. 1995, Wegner 1986). A recent study by Reagans et al. (2005) found that prior experience in working together increases the productivity of hospital teams. Hence, we expect that the greater the extent of shared prior experience among some or all members of the software team, the more the team members should have learned about working together, and the better their productivity.

Hypothesis 4B (4B). *A software group's current productivity in completing group work on a system is positively related to the average level of group members' shared experience from prior work with one another.*

3.2.3. Learning at the Organizational-Unit Level of Analysis. A prior history of working together on the same-system release will help developers in the organizational unit to build a shared understanding of the system, and a shared coding scheme that enhances the transfer and communication of knowledge (Zander and Kogut 1995). Having a shared coding scheme—in the form of codes, symbols, anecdotes, and rules about appropriate statements (Weber and Camerer 2003)—enables effective communication to take place between individuals in the same organizational unit. This shared experience will help developers in the unit to work together and coordinate with one another more efficiently, thus enabling them to become more productive in completing system releases.

Hypothesis 4C (H4C). *An organizational unit's current productivity in completing a system release is positively related to the average shared experience of the unit's developers from working together on prior system releases.*

4. Data and Methods

To evaluate our hypotheses, we analyzed an extensive archival data set covering more than 14 years of systems development work on a large product produced by the telecommunications company, dating from the beginning of its development process. In order to examine whether learning is occurring, it is imperative to have data about the product from the beginning of its development process, data about the characteristics of each unit of work completed, and data collected over a sufficiently long period such that learning can occur and can be exhibited (Argote 1999). The archival data set in this study meets all of these requirements and is thus particularly well suited to the examination of learning curves. The data were extracted from the version control and change management systems used by the organization to archive past versions of the product's source code. These systems capture a great deal of contextual information about each change made to the source code, about the contributions of each individual (who worked together with whom on which MRs), and about the complexity and size of the systems in the product.

4.1. Modification Requests (MRs) as the Unit of Analysis

The conventional form of the learning curve can be expressed as $y_t = ae^{\lambda t}K_t^{\alpha}x_{t-1}^{-\gamma}$, where y_t is the cost per unit to produce the xth unit in month t, a is the cost of producing the first unit, K represents inputs such as capital and labor, λ captures an exponential trend in output not explained by changes in the given inputs (often the control variables that are not part of the production function, such as time), x_{t-1} is the cumulative number of units produced as of month $t-1$, and γ is a parameter measuring the rate at which costs are reduced as cumulative output increases. The unit of analysis in the conventional form of the learning curve is the time period. The use of MR or system release as the unit of analysis enables us to incorporate independent variables to control for the size, complexity, and other factors expected to affect the completion of each MR or system release. Therefore we adopted the approach of Pisano et al. (2001) and Thornton and Thompson (2001, p. 1351) by analyzing "unit labor requirements" instead of monthly output rates. We thus define the MR and system release as the units in our analysis of individuals and groups, and organizational units, respectively. We analyzed data for 549,196 MRs (of which 518,971 are individual MRs and 30,225 are group MRs) in 2,282 system releases, completed by 5,123 developers over 14 years.

4.2. Dependent Variable: Software Development Productivity

Online Appendix A describes all of the variables used in the analyses. (All of the online appendices are available in the e-companion.)[1]

The dependent variable, productivity, is operationalized as the effort (number of labor hours in days) to complete the MR or system release. Unfortunately, our data do not include direct-effort measures for each MR. It is an onerous task for organizations to accurately record the effort of developers at such a fine-grained level. However, the data indicate on which MRs each developer was working at any time period. We thus adopted the effort-computation algorithm developed and validated by Graves and Mockus (1998), which estimates effort for each MR based on information available in a version control system. The effort for a system release is calculated by aggregating the effort for all of the MRs included in the release.[2]

4.3. Independent Variables

4.3.1. Prior Experience. We hypothesized that prior experience working on systems and working with other developers would affect current productivity. Consistent with the learning-curve literature, we measure prior systems experience as the number of work units, or MRs, completed prior to work on the current MR.[3] This experience measures focus on the learning-by-doing experience gained on the job, which occurs via the completion of work units. We distinguish between *same-system experience, related-systems experience*, and *unrelated-systems experience* in terms of the number of MRs completed on the same system, related systems, and unrelated systems, respectively, prior to the current MR. Evolving software tends to get more complex with each update, especially in large systems development efforts (Rajlich 2006). Due to the complexity of the product, there are associations between systems because of their shared responsibility in supporting new and existing features and functionalities (e.g., through sharing of common data). Thus, changes to a particular feature could affect more than one system, and the systems need to work together although each has been developed independently. For example, one system supports customer billing whereas another supports call recording. To implement a new feature that allows different calls to be charged at different rates, both systems are affected because they share common data that must be updated with the new feature. These two systems are therefore functionally more related to each other than to a system that handles packet-routing algorithms. We identify systems that are functionally related by examining the requests for adding new features or for making changes to existing features. If a request results in changes that affect two systems, then the two systems are functionally related as they likely share data or methods that need to be changed or added based on the request. We thus consider two systems to be related if they have at least the system-average number of requests impacting them both.[4]

Prior *experience working with other developers* is measured as the number of distinct developers with whom an individual has worked prior to the current MR.[5] We measured the *average shared experience* variables as the number of MRs (or system releases) completed by both members of each dyad in the group MR (or system release) across all systems, averaged across all dyads in the group (or system release).[6]

[1] An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

[2] Online Appendix B describes the details of the effort imputation.

[3] These experience measures only capture the experience of developers accumulated in one organization, and do not include experience accumulated outside the organization through training or work in other organizations.

[4] We conducted sensitivity analyses to examine how the results would change if we vary the criteria for defining related systems. Our results remain unchanged. Online Appendix C provides details of the analysis.

[5] To examine if it makes a difference to account for whether these other developers are experts or nonexperts, we conducted a sensitivity analysis operationalizing this variable as the number of different experts that an individual has worked with in the past. Using this alternative operationalization, however, did not provide as much explanatory power as the original model.

[6] This operationalization assumes that knowledge is relationship specific, or specific to each dyad (Reagans et al. 2005). An alternative team-experience measure counting the number of times that everyone on the team has worked together, "assumes that the team

4.3.2. Control Variables. Consistent with the learning-curve literature, we define learning to be the increase in productivity of developers as their experience increases. However, we need to differentiate between learning from experience and changes in productivity that may occur due to environmental changes or technological improvements that are independent of experience (Argote 1999, Argote et al. 1990, Thornton and Thompson 2001). Hence, we include a control variable *Time* at the start of each MR, which is defined as the number of months from the start of product development.

It is also important to control for other variables that could impact productivity. In software development, each unit of work differs in its characteristics. We thus include controls for MR and system-release characteristics. Based on prior research on software development, we identified several attributes of MRs that could impact productivity, including size, complexity, priority, and type. The effort devoted to the development of each software unit has been shown to increase with its size (Banker and Slaughter 1997). The *Size* of an MR is measured as the lines of code added, changed, or deleted.[7] Software complexity also affects the productivity of software development (e.g., Banker et al. 1998). The amount of coupling between files and components is a key indication of software complexity (Chidamber and Kemerer 1994). Tight coupling between software components make software enhancement more difficult. Any changes made to one part of the software that affect many other parts would require more effort to ensure correctness. Hence, we operationalize MR *Complexity* as the number of files that have to be changed to complete an MR (Herbsleb and Mockus 2003).

Priority and the type of work involved are two other important MR attributes. MR priority is determined by a change committee in the organization. There are four levels of MR *Priority*, coded into an ordinal scale ranging from 1 to 4: low (1), medium (2), high (3), and emergency (4). The higher the MR priority, the more urgent and important the work, and the more effort may be devoted to it. We also differentiate between new-feature development and feature maintenance. Maintenance sometimes requires a lot of effort to understand the code and identify the problem areas to be rectified, but in the end, may affect only one line of code. Thus, we include a binary variable *New Development*, which indicates whether an MR involves new development (1) or maintenance work (0).

4.4. Models at Different Levels of Analysis

We evaluate learning at the individual, group, and organizational-unit levels of analysis. At the individual level, we examine MRs that have been completed by only one individual ("individual MRs"); at the group level, we examine MRs that have been completed by more than one developer ("group MRs"); and at the organizational-unit level, we aggregate all of the MRs completed for a release by the organizational unit, and use system release as the unit of analysis. To specify our models at each level, we adopted the log-linear form of the learning-curve function (Yelle 1979), and define the effort for completing each work unit as a function of the MR or system-release characteristics and the experience variables.

4.4.1. Individual Level of Analysis. For individual MRs, we specify Equation (1) to test H1A, H1B, and H4A. The dependent variable is the effort per MR k in system i completed by individual j.

$$\begin{aligned}
&Ln(Effort\ Per\ MR_{ijk})\\
&\quad = \alpha_{0j} + \alpha_1 Ln(Size_{ijk}) + \alpha_2 Ln(Complexity_{ijk})\\
&\qquad + \alpha_3 New\ Development_{ijk} + \alpha_4 Priority_{ijk} + \alpha_5 Time_{ijk}\\
&\qquad + \alpha_{6j} Ln(Individual\ Same\text{-}System\ Experience_{ijk})\\
&\qquad + \alpha_{7j} Ln(Individual\ Related\text{-}Systems\ Experience_{ijk})\\
&\qquad + \alpha_{8j} Ln(Individual\ Unrelated\text{-}Systems\ Experience_{ijk})\\
&\qquad + \alpha_{9j} Ln(Individual\ Experience\\
&\qquad\qquad with\ Other\ Developers_{ijk})\\
&\qquad + \alpha_{10-97} System_i + u_j + \epsilon_{ijk}. \qquad (1)
\end{aligned}$$

To test H1A and H1B, we compared the relative effects of individual's prior same-system experience (α_6), individual's prior related-systems experience (α_7), and individual's prior unrelated-systems experience (α_8) on the effort required to complete the current MR. The coefficients, α_{6j}, α_{7j}, and α_{8j}, constitute the learning indices from which we can derive the rate at which individuals reduce the effort required per MR based on their experience gained from prior work on the same, related, and unrelated systems, respectively (Yelle 1979). To test H4A, we examined the impact of the individual's prior experience working with other developers on the effort required for the current MR (α_9).

Our data for the individual MRs have a nested structure: MRs are nested within individuals, and

is only as experienced as the least experienced pair" (Reagans et al. 2005, p. 877). We thus conducted a sensitivity analysis operationalizing the average shared experience as the number of times that everyone on the team has worked together. Using this alternative operationalization, however, did not provide as high an explanatory power as the original operationalization of average shared experience. These results therefore suggest that in temporary teams, such as software groups, coordination appears to be relationship specific, rather than specific to the team as a whole.

[7] Modifying a line of code is equivalent to adding a line of code and deleting a line of code.

individuals are nested within one or more systems. We specified the systems as fixed effects, and individuals as random effects.[8] Random effects can be estimated not only for the intercept but also for the slopes. Random slope coefficients are appropriate if the estimated coefficients of the independent variables differ across units. The literature on individual learning suggests that not all individuals learn at the same rate. Thus, we model differences in learning rates across individuals by estimating random coefficients for each individual for the different types of experience, allowing individual learning rates α_{6j}, α_{7j}, α_{8j}, and α_{9j} to vary across individuals. The model thus effectively becomes an individual growth model (Singer 2002).

4.4.2. Group Level of Analysis. For group MRs, we specify Equation (2) to test H2A, H2B, and H4B. The dependent variable is the effort per group MR k in system i.

$$\begin{aligned}
&Ln(Effort\ Per\ MR_{ik})\\
&= \beta_0 + \beta_1 Ln(Size_{ik}) + \beta_2 Ln(Complexity_{ik})\\
&\quad + \beta_3 Ln(Number\ of\ Developers_{ik})\\
&\quad + \beta_4 New\ Development_{ik} + \beta_5 Priority_{ik} + \beta_6 Time_{ik}\\
&\quad + \beta_7 Ln(Avg.\ Group\ Same\text{-}System\ Experience_{ik})\\
&\quad + \beta_8 Ln(Avg.\ Group\ Related\text{-}Systems\ Experience_{ik})\\
&\quad + \beta_9 Ln(Avg.\ Group\ Unrelated\text{-}Systems\ Experience_{ik})\\
&\quad + \beta_{10} Ln(Avg.\ Shared\ Experience_{ik})\\
&\quad + \beta_{11-88} System_i + \epsilon_{ik}. \qquad (2)
\end{aligned}$$

In this analysis, the dependent variable is the effort required to complete each group MR, or the sum of the imputed effort expended by all individuals working on the group MR. To test H2, we compared the effects of average group same-system experience (or the average number of MRs for the current system completed by individuals working on the group MR, prior to its start) to the effects of average group experience in related and unrelated systems (or the average number of MRs for related and unrelated systems completed by individuals working on the group MR, prior to its start). The coefficients, β_7, β_8, and β_9, constitute the learning indices from which we can derive the rate at which groups reduce the effort required per MR based on their experience gained from the same, related and unrelated systems, respectively. To test H4B, we examined the effects of average shared experience or the number of MRs completed by both members of each dyad included in the group MR prior to the start of the MR, averaged across all dyads in the group.

Our data for group MRs also have a nested structure as group MRs are nested within systems. For consistency, and for similar considerations as those for individual MRs, we continue to consider systems as fixed effects. The control variables for each group MR are similar to those in the analysis of individual MRs, except for the inclusion of an additional control variable—the number of developers involved in each group MR. As the group size increases, we expect more effort to be expended on coordination (Espinosa 2002, Kraut and Streeter 1995), thus increasing the effort required for group MRs.

4.4.3. Organizational-Unit Level of Analysis. For system releases, we specify Equation (3) to test H3 and H4C. The dependent variable is the effort per release r for system j:

$$\begin{aligned}
&Ln(Effort\ Per\ System\ Release_{jr})\\
&= \gamma_0 + \gamma_1 Ln(Size_{jr}) + \gamma_2 Ln(Complexity_{jr})\\
&\quad + \gamma_3 Ln(\%\ New\ Group\ MRs_{jr})\\
&\quad + \gamma_4 Ln(\%\ Maintenance\ Individual\ MRs_{jr})\\
&\quad + \gamma_5 Ln(\%\ Maintenance\ Group\ MRs_{jr})\\
&\quad + \gamma_6 Ln(Avg.\ No.\ of\ Developers\ per\ MR_{jr})\\
&\quad + \gamma_7 Ln(Avg.\ MR\ Priority_{jr}) + \gamma_8 Time_{jr}\\
&\quad + \gamma_9 Ln(Avg.\ Unit\ Same\text{-}System\ Experience_{jr})\\
&\quad + \gamma_{10} Ln(Avg.\ Unit\ Related\text{-}Systems\ Experience_{jr})\\
&\quad + \gamma_{11} Ln(Avg.\ Unit\ Unrelated\text{-}Systems\ Experience_{jr})\\
&\quad + \gamma_{12} Ln(Avg.\ Shared\ Experience_{jr})\\
&\quad + \gamma_{13-79} System_j + \epsilon_{jr}. \qquad (3)
\end{aligned}$$

The dependent variable is defined as the effort required to complete each release, measured as the sum of the imputed effort of all MRs included in the system release. All of the experience variables are similar to those defined at the group level of

[8] An effect is fixed if the levels in the study represent all possible levels of the factor about which inference is to be made (Littell et al. 1996). A fixed-effects formulation assumes that differences across units can be best captured by estimating a different intercept term for each unit and is appropriate if the units differ in their average level of the outcome variable. We model systems as fixed effects because it is likely that systems differ in their average levels of productivity. Moreover, modeling the system as random effects would assume that the system effects are uncorrelated with other regressors (Greene 2003), which seems highly unlikely, as some systems are bound to be more complex or larger than others. In contrast, a random-effects formulation assumes that the levels of the factor used in the study represent a random sample of a larger set of potential levels (Greene 2003) and that differences across units can be best captured by estimating a unique error term for each unit. As we are trying to generalize the learning among individuals in our sample to other software developers, we define the individual level as random.

analysis, except that the experience variables are averaged across all the individuals in the organizational unit working on the current system release. To test H3, we compared the effects of average unit same-system experience (or the average number of MRs for the current system completed by individuals working on the release, prior to its start) to the effects of average unit experience in related and unrelated systems (or the average number of MRs for related and unrelated systems completed by individuals working on the current release, prior to its start). The coefficients, γ_9, γ_{10}, and γ_{11}, constitute the learning indices from which we can derive the rate at which the organizational units reduce the effort required per release based on the experience gained from the same, related, and unrelated systems, respectively. To test H4C, we examined the effects of average shared experience or the number of system releases which both members of a dyad involved in the system release had completed prior to the start of the system release, averaged across all dyads in the organizational unit completing the system release.

Our data for the system releases has a nested structure as each release is nested within a system. We continue to consider systems as fixed effects for reasons given earlier. The control variables for each system release are similar to those for individual and group MRs. The size of each release refers to the sum of the size of all MRs in the system release. The release complexity refers to the average complexity of all the MRs included in the system release. We also included a control for the number of developers per MR for the system release, as this variable provides an indication of the extent of coordination required in completing the system release. Average priority refers to the average priority of all the MRs in the system release. In addition, we observed that system releases that tend to have a greater percent of new development also tend to have a greater percent of group MRs. This appears to indicate that new-feature developments tend to require a group of developers to work on them. Due to this interaction between new development and individual versus group MRs, we classified all MRs into four categories for each release: (1) new individual MRs; (2) new group MRs; (3) maintenance individual MRs; and (4) maintenance group MRs. Using new individual MRs as the base category, we included three variables that represent the percentage of MRs in each of the three remaining categories for every system release.

5. Estimation and Results

Due to the nested nature of the data, we estimated each of our models using multilevel modeling[9] (Ang et al. 2002, Bryk and Raudenbush 1992, Hofmann 1997), which allows us to model the fixed and random effects, and accommodate the nesting of repeated measures within individuals and systems. Feasible generalized least squares (FGLS) is used to estimate the coefficients, and the Newton-Raphson algorithm (Lindstrom and Bates 1988) is used to derive maximum likelihood estimates of the variance-covariance components for the residuals.[10] Multilevel modeling affords considerable flexibility as the number of observations per system or per individual may vary, and the time variable can be continuous rather than fixed intervals. We conducted a number of specification checks for the models, including specifying the system level as random instead of fixed, and the individual level as fixed instead of random. Results are consistent with our original specifications, although our original specifications yield the best model fit. To correct for autocorrelation, we specified first-order autoregressive (AR(1)) as the within-subject covariance structure for repeated measures (covariance structure $= \sigma^2\rho^{|i-j|}$).[11] We also checked for multicollinearity,[12] and found no evidence of multicollinearity problems. Finally, to examine the homogeneity of variance assumption, we modeled both within-subject and between-subject error covariance structures, and found no evidence of heteroskedasticity problems.[13]

5.1. Individual Level Results

Table 1 presents the results of the estimation for individual MRs. We first established that there is significant variation in the effort per MR among individual MRs to be explained by estimating only an

[9] Online Appendix D shows the descriptive statistics and intercorrelations for variables for all three levels of analysis.

[10] We used PROC MIXED in the SAS statistical package (SAS Institute 2000, Singer 2002) and full maximum likelihood because it yields log-likelihood numbers that are used to evaluate the incremental fit across nested models. Similar results are obtained using restricted maximum likelihood (also implemented with a Newton-Raphson algorithm).

[11] We compared the AR(1) within-subject error structure to the default error structure where repeated observations within subject are assumed to be independent, and the spatial powers (SP(POW)) error structure for nonequally-spaced time series data. Consistent results were obtained in all three analyses. The AR(1) error structure was chosen because it provides the best fit to our data.

[12] We repeated our analyses in using ordinary least squares (OLS) at all three levels of analysis, and found that the highest condition index was 22.37, and variance inflation factor (VIF) values were all below 3. We examined the pairwise correlations between the variables at each level of analysis and found that the correlations are relatively modest—no higher than 0.55. Robustness checks also show that there are no multicollinearity problems at all three levels of analysis. Online Appendix E provides more details of the robustness checks.

[13] This approach does not make any strong assumptions about the nature of the heteroskedasticity (Greene 2003).

Table 1 Results for Individual MRs (Dependent Variable: *Ln Effort Per Individual MR***)**

Variables	Model 1	Model 2	Model 3	Model 4
Intercept (α_0)	−1.986*** (0.001)	−1.812*** (0.017)	−1.891*** (0.016)	−1.768*** (0.019)
Ln Size (α_1)		0.050*** (0.000)	0.050*** (0.000)	0.050*** (0.000)
Ln File Complexity (α_2)		0.066*** (0.001)	0.068*** (0.001)	0.068*** (0.001)
New Development (α_3)		−0.267*** (0.002)	−0.266*** (0.002)	−0.273*** (0.002)
Priority (α_4)		0.090*** (0.001)	0.091*** (0.001)	0.092*** (0.001)
Start Month (α_5)		−0.003*** (0.000)	−0.002*** (0.000)	−0.001*** (0.000)
Ln Individual Same-System Experience (α_{6j})			−0.034*** (0.000)	−0.033*** (0.001)
Ln Individual Related-Systems Experience (α_{7j})			−0.025*** (0.000)	−0.026*** (0.001)
Ln Individual Unrelated-Systems Experience (α_{8j})			−0.007*** (0.000)	−0.009*** (0.001)
Ln Individual Experience Working with Others (α_{9j})			−0.016*** (0.000)	−0.013*** (0.001)
Deviance (−2 log likelihood)	1,526,729	791,857	763,571	685,662
Within-individual variance	1.1096	1.0723	0.8978	0.5165
Deviance difference (Δdev)		734,873	28,286	77,909
Proportion of within-individual variance explained (%)		3.36	19.09	53.45

Notes. (1) $^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$ for all analyses, and standard errors of beta coefficients are presented in parentheses for Tables 1, 2, and 3. *P*-values are based on two-sided *t*-tests for the significance of the coefficients. (2) Number of observations (no. of MRs) = 518,971; number of subjects for random effects (no. of individuals) = 5,123; number of systems included in fixed-effects = 89, degrees of freedom (df) = 510,000 for estimation of coefficients. (3) Within-subject correction for AR(1) error structure included for Models 2–4. (4) Individual random effects are included for Model 4, and all random effect intercepts and coefficients are allowed to covary with one another (i.e., the unstructured covariance structure is adopted for the random effects).

intercept term (Model 1 in Table 1).[14] Model 2 adds the control variables, Model 3 adds the experience variables, and Model 4 allows the individual intercepts and experience slopes to vary. The proportion of variance explained from adding each set of determinants to the models was calculated. We examined the significance of the incremental variance explained by examining the differences between the deviance statistics (Δdev) for each pair of nested models. Δdev is twice the negative log likelihood, and has a chi-square distribution with the difference in number of parameters between models to be estimated as the degrees of freedom. The incremental variances explained by Models 2, 3, and 4 are all significant ($p < 0.001$).

The results indicate that effort per individual MR decreases with increasing individual same-system experience (Equation (1), $\alpha_6 = -0.033$, $p < 0.001$), individual related-systems experience (Equation (1), $\alpha_7 = -0.026$, $p < 0.001$), and individual unrelated-systems experience (Equation (1), $\alpha_8 = -0.009$, $p < 0.001$).[15] To test H1A and H1B, we compared the regression coefficients of the same-system, related-systems and unrelated-systems experience variables. We found that specialized experience within a system has a greater impact on individual productivity than diverse experience in related ($t = 6.22$, $p < 0.001$) and unrelated systems ($t = 23.16$, $p < 0.001$), providing support for H1A. Further, experience in related systems has a greater impact on individual productivity than experience in unrelated systems ($t = 15.67$, $p < 0.001$), which supports H1B. We also found support for H4A, that is, the effort per MR decreases as the number of different developers the individual has worked with in prior MRs increases (Equation (1), $\alpha_9 = -0.013$, $p < 0.001$).

Our estimates for the random effects reflect the ability of individuals to learn from different types of experience. The variances of the random-effects coefficients reveal the variability in the initial productivity (intercepts) and in learning rates (slopes) from different types of experience variables (Singer 2002). The significance of the variance of the random-effects coefficients estimated at the individual level of analysis implies that developers differ in their initial productivity, and that they learn at different rates.

5.2. Group Level Results

Table 2 presents the results of the estimation for group MRs. Similar to the above analysis, we first established that there is significant variation in effort per group MR to be explained in Model 1.[16] Models 2 and

[14] Results indicate that there is significant variation in effort per individual MR to be explained, and that more variation in effort per MR is between MRs completed by an individual than between MRs completed by different individuals (57.23% versus 42.77%), and that most of the variation in effort per MR is between MRs in a system than between systems (81.17% versus 18.83%).

[15] The coefficients suggest that adding one more same-system MR to a developer's experience would allow the developer to improve his or her productivity by 0.290 hours per MR, on average. Adding one more related-system (unrelated-system) MR to a developer's experience would allow the developer to improve his or her productivity by 0.232 (0.079) hours per MR, on average.

[16] Results indicate that there is significant variation in effort per group MR to be explained and that 73.53% of the variance in effort per group MR is between MRs within a system, whereas 26.47% of the variance is between systems.

Table 2 Results for Group MRs (Dependent Variable: *Ln Effort Per Group MR***)**

Variables	Model 1	Model 2	Model 3
Intercept (β_0)	0.194*** (0.007)	−1.337*** (0.043)	−1.761*** (0.039)
Ln Size (β_1)		0.043*** (0.001)	0.034*** (0.001)
Ln File Complexity (β_2)		0.433*** (0.003)	0.446*** (0.003)
Ln Number of Developers in Group MR (β_3)		0.932*** (0.014)	1.119*** (0.013)
New Development (β_4)		0.080*** (0.008)	0.106*** (0.007)
Priority (β_5)		−0.043*** (0.003)	−0.011*** (0.003)
Start Month (β_6)		−0.003*** (0.000)	−0.001*** (0.000)
Ln Average Group Same-System Experience (β_7)			−0.024*** (0.001)
Ln Average Group Related-Systems Experience (β_8)			−0.034*** (0.001)
Ln Average Group Unrelated-Systems Experience (β_9)			−0.016*** (0.001)
Ln Average Shared Group Experience (β_{10})			−0.027*** (0.001)
Deviance (−2 log likelihood)	94,847.0	55,286.9	49,802.6
Within-system variance	1.3501	0.3656	0.3047
Deviance difference (ΔDev)		39,560	5,484
Proportion of within-system variance explained (%)		72.92	77.43

Notes. (1) number of observations (no. of group MRs) = 30,225; number of systems included in fixed-effects = 79, df = 30,000. (2) Within-subject correction for AR(1) error structure included for Models 2 and 3.

3 were estimated to evaluate the proportion of variance explained from adding the control variables and the experience variables, respectively. All incremental variances explained by Models 2 and 3 are significant (at $p < 0.001$).

Our results indicate that effort per group MR decreases with increasing average same-system experience (Equation (2), $\beta_7 = -0.024$, $p < 0.001$), average related-systems experience (Equation (2), $\beta_8 = -0.034$, $p < 0.001$), and average unrelated-systems experience (Equation (2), $\beta_9 = -0.016$, $p < 0.001$).[17] We tested H2 by comparing the regression coefficients of the same-system, related-systems and unrelated-systems experience variables. Our results show that average group experience in related systems has a greater impact on group productivity than average group experience in the same system ($t = 6.12$, $p < 0.001$) and average group unrelated-systems experience ($t = 16.68$, $p < 0.001$). Average group experience in the same system, on the other hand, had a greater impact on group productivity than average group unrelated-systems experience ($t = 5.70$, $p < 0.001$). Hence, H2 was only partially supported, as the results show that although related-system experience had a greater impact on group productivity than same-system experience, unrelated-system experience had less impact on group productivity than same system experience. In addition, we found support for H4B, as our results indicate that effort per group MR decreases with increasing average shared experience of group members (Equation (2), $\beta_{10} = -0.027$, $p < 0.001$).

5.3. Organizational-Unit Level Results

Table 3 presents the results for the estimation of system releases. We first established that there is significant variation in effort per system release to be explained.[18] Models 2 and 3 were estimated to evaluate the proportion of variance explained from adding the control and experience variables, respectively. All incremental variances explained by Models 2 and 3 are significant ($p < 0.001$).

Our results suggest that the average same-system experience (Equation (3), $\gamma_9 = -0.014$, $p > 0.10$) and the average unrelated-system experience (Equation (3), $\gamma_{11} = -0.010$, $p > 0.10$) of individuals in the organizational unit completing the release has no significant effect on the effort per release. The average related-system experience of developers involved in completing the release, on the other hand, significantly decreases the effort per release (Equation (3), $\gamma_{10} = -0.053$, $p < 0.001$).[19] To test H3, we compared the regression coefficients of the same-system experience variable to the related-systems and unrelated-systems experience variables. Our results show that average

[17] The coefficients suggest that adding one more same-system MR to a group's average experience would improve group productivity by 1.265 hours per group MR. Adding one more related-system MR to a group's average experience would improve group productivity by 1.798 hours per group MR, whereas adding one more unrelated-system MR to a group's average experience would only improve group productivity by 0.833 hours per group MR.

[18] Results indicate that there is significant variation in effort per system release to be explained and that 74.18% of the variance in effort per system release is between releases within a system, whereas 25.82% of the variance is between systems.

[19] The coefficients suggest that adding one more same-system MR to a system release's average organizational-unit experience would improve system-release productivity by 1.43 hours per system-release. Adding one more related system MR to a system release's average experience would improve system-release productivity by 5.47 hours per release, whereas adding one more unrelated system MR to a system release's average experience would only improve system-release productivity by 1.05 hours per release.

Table 3 Results for System Releases (Dependent Variable: *Ln Effort Per System Release*)

Variables	Model 1	Model 2	Model 3
Intercept (γ_0)	2.779*** (0.045)	1.088*** (0.264)	1.016*** (0.262)
Ln Size (γ_1)		0.415*** (0.010)	0.400*** (0.009)
Ln Average File Complexity (γ_2)		−0.334*** (0.039)	−0.297*** (0.038)
Ln Percent New Group MRs (γ_3)		0.077*** (0.006)	0.064*** (0.006)
Ln Percent Maintenance Individual MRs (γ_4)		0.048*** (0.008)	0.051*** (0.008)
Ln Percent Maintenance Group MRs (γ_5)		0.096*** (0.006)	0.085*** (0.006)
Ln Avg. No. of Developers Per MR (γ_6)		−0.706*** (0.089)	−0.712*** (0.087)
Ln Avg. Priority (γ_7)		0.029 (0.017)	0.033* (0.016)
Start Month (γ_8)		−0.000 (0.000)	−0.000 (0.000)
Ln Average Unit Same-System Exp. (γ_9)			−0.014 (0.012)
Ln Average Unit Unrelated-Systems Exp. (γ_{11})			−0.010 (0.009)
Ln Average Shared System Release Exp. (γ_{12})			0.056*** (0.005)
Deviance (−2 Log Likelihood)	9,966.4	6,138.4	6,017.6
Within-system variance	4.6092	0.8649	0.8209
Deviance difference (ΔDev)		3,828	121
Proportion of within-system variance explained (%)		81.24	82.19

Notes. (1) Number of observations (no. of system releases) = 2,282; number of systems included in fixed-effects = 68, df = 2202. (2) Within-subject correction for AR(1) repeated measures error structure included for Models 2 and 3.

unit experience in related systems had a significantly greater impact on system-release productivity than average unit experience in the same system ($t = 2.24$, $p < 0.05$) and average unit unrelated-system experience ($t = 3.17$, $p < 0.01$). There were no significant differences in the impact of average unit same-system experience and average unit unrelated-systems experience on system-release productivity ($t = 0.22$, $p > 0.10$). These results provided partial support for H3, as the results show that although related-system experience had a greater impact on productivity than same-system experience, unrelated-system experience had the same impact on productivity as same-system experience. We also found that H4C is not supported as the effort per release increased with the average shared system release experience of individuals in the organizational unit (Equation (3), $\gamma_{11} = 0.056$, $p < 0.001$).

6. Discussion

In this section, we discuss our results, and provide interpretations and support for our findings by drawing on insights from interviews with six key informants in the organization. The key informants include three software developers and three quality-assurance personnel. These interviewees were selected to provide the perspectives of both developers and employees involved in process improvement. Most interviewees have worked on this product for at least ten years; thus they were able to provide insights into the institutional context and history of the development process for the product.

6.1. Learning from Individual Experience

At the individual level, specialized experience has the greatest impact on individual productivity (H1A), probably because of the effort to learn a new system, as highlighted by one interviewee: "Every time you do something, there's a learning curve, unless you've seen it before" (Developer A). Interviews with developers further bolstered our argument that experience from working on the same system enables developers to gain familiarity with the system domain and increases understanding of the structure and architecture of the components and files, as well as the code. A developer highlighted how experience in a system helped him to understand the system as a whole:

> The individuals become productive when they learn all the aspects of [the system].... If you're very new to it, then you may have difficulty figuring out, if you make this change, how does it impact everyone else, or how does this really relate to what they're doing over there. After a while, you know all of that. I know that I shouldn't change this, because it will have an impact over there, but if I do it this way, it will be much more isolated." (Developer B)

We further found that prior experience in related systems had a more significant effect than experience in unrelated systems on individual productivity (H1B). This shows that developers' prior experience from working on related systems transfers across systems more effectively than their experience in unrelated systems, to improve their productivity in work on the current system.

At the group level of analysis, we found that diverse experience in related systems had a more significant impact than specialized experience in the same system (H2). This confirms prior research about the role of individual diversity in improving group performance. Although Bunderson and Sutcliffe (2002) found a positive relationship between individual functional diversity and team performance, the managers in their sample had a fairly narrow range of functional experience. The question that remained unanswered, therefore, was whether the finding was applicable for groups where individuals had a wide range of experiences. Our study thus provides some insights into this question. Our results show that experience in unrelated systems appeared to have less impact on productivity than experience in the same system and in related systems. This is perhaps because of the difficulties in integrating experiences across a wide range of systems, and because experience in unrelated systems is typically less relevant than experience in the same and related systems. These results suggest that it is important for managers to pay attention to the extent of the diversity in experience that individuals bring into a team, as too much diversity in experience may not necessarily be good for the team.

At the organizational-unit level of analysis, we found that only experience in related systems improved system-release productivity, whereas average unit experience in the same system or in unrelated systems had no effect on this type of productivity. Diverse experience in related systems is important for organizational units for several reasons. First, organizations gain greater flexibility from being able to move developers to different systems based on the differing needs of the organizational units. As highlighted by an interviewee, the ability to move people to different systems provides the flexibility to cater to differing levels of demand in different systems:

> Many [developers] have experience in more than one [system]. This is so support can be given to the active development on the effect of changes on other systems. (Quality manager D)

Second, increased absorptive capacity from greater breadth of knowledge may also explain why diverse experience in related systems improves the productivity of organizational units. Although diverse experience may be costly for each individual, the organizational unit as a whole is able to benefit from the flexibility in task assignment and from having experts with an overall view of the related parts of the product:

> It is useful to have these experts around, who have a good view of the software, and can give a deciding view on certain more tricky problems. (Quality manager D)

Our results, however, show that it is the investment in diverse experience in related systems that provide the greatest payoff for organizational units. Accumulating diverse experience in unrelated systems is not an effective strategy, perhaps because too much diversity in experience makes it difficult for the organizational unit to integrate the experience across all individuals working on the system release. It could also be that knowledge about unrelated systems is less relevant to the task at hand.

In summary, we found that the relative importance of the different types of experience differs *across levels of analysis*. At the individual level, specialized experience within the same system has the greatest impact on individual productivity. However, as we progress from the individual, to the group, and organizational unit levels of analysis, diverse experience in related systems plays a larger role in improving the productivity of groups and organizational units. Diverse experience in unrelated systems, however, remains the factor having the least influence on productivity at all three levels of analysis. Our results provide guidance to organizations about how they can balance between the need for exploitation and exploration. At the individual level, exploitation appears to be the most effective strategy, as management can capitalize on the specialized experience gained by developers to improve productivity at the individual level. With larger and more complex MRs and system releases, however, our results highlight the increasingly important role of having an exploration strategy, namely by increasing the diversity of developers' experience in related systems. Such diverse experience will enable developers to work more effectively with one another, improve the absorptive capacity of the organizational-unit, and provide more flexibility to the organizational unit in allocating work to developers, thus improving group and organizational-unit productivity. Nevertheless, managers must be cognizant of the costs involved in their exploration strategy, as our results for unrelated-system experience shows that the productivity benefits of providing developers with experience in a large number of unrelated systems may not be substantial.

6.2. Learning from Working with Others

Our results also emphasize the learning opportunities from working with others in software development. At the individual level, our results suggest that individual developers do learn from other developers, as individuals exhibit greater improvements in productivity when they work with a greater number of developers (H4A). Our interviewees also highlighted the importance of having more experienced developers transfer their knowledge to new developers:

> As new people come, and work in an area they're not familiar with, I can help them get on board quicker. I help review, discuss things with them, so having the experts in each system helps to move people forward as you bring in people with different skills. (Developer A)

Prior experience working with one another also significantly improves the productivity of developers working together on a group MR (H4B). This finding highlights the advantages of putting individuals who have experience working with one another on the same team in software development. This hypothesis also found support in the interviews we conducted. A developer, for example, highlighted the importance of prior working relationships in understanding each other's work:

> Information transfer is simple for engineers who have worked together for a period of time since common similarities can be found. (Developer C)

Finally, the results show that greater shared experience among developers in the same system release actually decreases the productivity of the organizational unit in completing system releases, rather than improve the productivity as we had expected (H4C). This result highlights that although sharing of experience at the group level will increase productivity by improving coordination and influencing the ease of knowledge sharing, sharing of experience at the organizational unit actually decreases productivity. One possible reason is that at the organizational-unit level, coordination is required on a larger scale and is more formally done. Work may be more structured and managed in a more centralized approach, and explicit rather than implicit coordination mechanisms may be leveraged. Thus, the importance of having shared knowledge developed through working with one another becomes less important in affecting the productivity of system releases, compared to group MRs. Moreover, too much sharing of experience at the organizational-unit level may lead to the entrenchment of certain practices or habits that may be difficult to change (Brown and Eisenhardt 1995). For example, one of our interviewees highlighted how individuals might be unwilling to deviate from past practices:

> Following a regimented path leads to people making the same mistakes since they don't want to deviate from the practices of past developments that they have agreed upon. (Developer B)

6.3. Limitations

The results of this study should be interpreted with several limitations in mind. First, the learning-curve approach is an effective method to examine whether there are any productivity improvements in organizations due to the accumulation of experience, but the method does not examine the specific mechanisms by which experience affects productivity. Thus, it is important for future research to examine the different learning mechanisms adopted by organizations and factors influencing their effectiveness. Second, by examining productivity as the dependent variable, we focused on a very important but single aspect of learning—learning to complete MRs with the least amount of effort, taking into consideration the complexity, size, and other aspects of the MRs. An important direction for future research is to examine other aspects of learning in software development, such as learning about quality and process improvement. Finally, we examined learning in the context of incremental software development. A different scale of learning may be applicable in purely new development, and it is therefore important for future research to examine whether our results apply in that context.

7. Conclusions

Our study extends the literature on organizational learning by discerning the improvement in productivity that is attributable to different kinds of experience at the individual, group and organizational-unit levels of analysis. We also provide insights into the relative value of specialized versus diversified experience for learning. This is important, because in leveraging experience, managers need to weigh the gains from specialized experience versus diverse experience in influencing productivity. Our study contributes to the limited number of studies examining the balance of exploitation and exploration across multiple levels of analysis (Gupta et al. 2006), and provides suggestions about how software development organizations can effectively balance between exploitation and exploration strategies. At the individual level, exploitation appears to be an effective strategy, as management can capitalize on the specialized experience gained by developers to improve individual productivity. To handle larger and more complex MRs and system releases at the group and organizational-unit levels, organizations may need to also adopt an exploration strategy to provide developers with diverse experience in related systems. This influences the productivity of groups and organizational units by facilitating developers' work in teams, and increasing the absorptive capacity of the organizational unit. Nevertheless, organization units must be cognizant of the costs involved in their exploration strategy, as our results show that the benefits of providing developers with experience in unrelated systems are unlikely to be substantial. In addition, our study shows that working with others provides developers with valuable opportunities to learn from others and to learn to work with others.

Our study also contributes by showing that the learning curve is salient in knowledge work and by quantifying the extent to which learning improves performance in knowledge work activities like software development. Understanding whether individuals, groups, and organizations can leverage prior experience to improve productivity in knowledge work has important implications for hiring, training, and organization of work. Our analyses reveal that the highest rate of learning from individual experience in our context was about 21%,[20] lower than the learning rate of 80% found in manufacturing studies (Argote 1999). This suggests that experience has a substantially larger impact on productivity in manufacturing compared to knowledge work. Further, because software development tasks are somewhat more structured and repetitive than other knowledge-work tasks, the learning rate in those tasks could be even lower.

8. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

Acknowledgments

The authors thank Linda Argote, Jim Herbsleb, Sara Kiesler, Ann Majchrzak, Audris Mockus, and Denise Rousseau, for their help and advice. The authors also gratefully acknowledge the valuable comments of participants in research seminars at Carnegie Mellon University, the University of Oklahoma, the University of Michigan, and Georgia Tech University, where earlier versions of this paper were presented. A prior version of this paper was also presented at the Academy of Management Meeting in 2003.

References

Ang, S., S. A. Slaughter, K. Y. Ng. 2002. Human capital and institutional determinants of information technology compensation: Modeling multilevel and cross-level interactions. *Management Sci.* **48**(11) 1427–1445.

Argote, L. 1999. *Organizational Learning: Creating, Retaining, and Transferring Knowledge*. Kluwer, Norwell, MA.

Argote, L., S. L. Beckman, D. Epple. 1990. The persistence and transfer of learning in industrial settings. *Management Sci.* **36**(2) 140–154.

Banker, R. D., S. A. Slaughter. 1997. A field study of scale economies in software maintenance. *Management Sci.* **43**(12) 1709–1725.

Banker, R. D., G. B. Davis, S. A. Slaughter. 1998. Software development practices, software complexity, and software maintenance performance: A field study. *Management Sci.* **44**(4) 433–450.

Basili, V. R., G. Caldiera. 1995. Improve software quality by reusing knowledge and experience. *Sloan Management Rev.* **37**(1) 55–64.

Brooks, F. P. 1995. *The Mythical Man Month: Essays on Software Engineering*. Prentice-Hall, Reading, MA.

Brown, S. L., K. M. Eisenhardt. 1995. Product development: Past research, present findings, and future directions. *Acad. Management Rev.* **20**(2) 343–378.

Bryk, A. S., S. W. Raudenbush. 1992. *Hierarchical Linear Models: Applications and Data Analysis Methods*, 1st ed. Sage, Newbury Park, CA.

[20] This *progress ratio* (see Argote 1999) is calculated by multiplying the maximum learning rate by the mean number of individual and group MRs completed per month. The learning rate for the system release is lower (less than 10%).

Bunderson, J. S., K. M. Sutcliffe. 2002. Comparing alternative conceptualizations of functional diversity in management teams: Processes and performance effects. *Acad. Management J.* **45**(5) 873–893.

Chidamber, S. R., C. F. Kemerer. 1994. A metrics suite for object oriented design. *IEEE Trans. Software Engrg.* **20**(6) 476–493.

Cohen, W. M., D. A. Levinthal. 1990. Absorptive capacity: A new perspective on learning and innovation. *Admin. Sci. Quart.* **35**(1) 128–152.

Crossan, M. M., H. W. Lane, R. E. White. 1999. An organizational learning framework: From intuition to institution. *Acad. Management Rev.* **24**(3) 522–537.

Darr, E. D., L. Argote, D. Epple. 1995. The acquisition, transfer, and depreciation of knowledge in service organizations: Productivity in franchises. *Management Sci.* **41**(11) 1750–1762.

Ellis, H. 1965. *The Transfer of Learning*. Macmillan Company, New York.

Espinosa, J. A. 2002. Shared mental models and coordination in large-scale, distributed software development. Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Faraj, S., L. Sproull. 2000. Coordinating expertise in software development teams. *Management Sci.* **46**(12) 1554–1568.

Fichman, R. G., C. F. Kemerer. 1997. The assimilation of software process innovations: An organizational learning perspective. *Management Sci.* **43**(10) 1345–1363.

Garud, R., A. Kumaraswamy. 2005. Vicious and virtuous circles in the management of knowledge: The case of InfoSys Technologies. *MIS Quart.* **29**(1) 9–33.

Graves, T. L., A. Mockus. 1998. Inferring change effort from configuration management databases. *5th IEEE Internat. Sympos. Software Metrics, Bethesda, MD,* http://www.mockus.us/papers/effort/.

Greene, W. H. 2003. *Econometric Analysis*, 5th ed. Prentice-Hall, Upper Saddle River, NJ.

Gupta, A., K. G. Smith, C. E. Shalley. 2006. The interplay between exploration and exploitation. *Acad. Management J.* **49**(4) 693–706.

Herbsleb, J. D., A. Mockus. 2003. An empirical study of speed and communication in globally-distributed software development. *IEEE Trans. Software Engrg.* **29**(6) 481–494.

Hofmann, D. A. 1997. An overview of the logic and rationale of heirarchical linear models. *J. Management* **23**(6) 723–744.

Holmqvist, M. 2004. Experiential learning processes of exploitation and exploration within and between organizations: An empirical study of product development. *Organ. Sci.* **15**(1) 70–81.

Katz, R. 1982. The effects of group longevity on project communication and performance. *Admin. Sci. Quart.* **27**(1) 81–104.

Kraut, R. E., L. A. Streeter. 1995. Coordination in software development. *Comm. ACM* **38**(3) 69–81.

Levitt, B., J. G. March. 1988. Organizational learning. *Annual Rev. Sociol.* **14** 319–340.

Liang, D. W., R. Moreland, L. Argote. 1995. Group versus individual training and group performance: The mediating role of transactive memory. *Personality and Soc. Psych. Bull.* **21**(4) 384–393.

Lindstrom, M. J., D. M. Bates. 1988. Newton-Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *J. Amer. Statist. Assoc.* **83**(404) 1014–1022.

Littell, R. C., G. A. Milliken, W. W. Stroup, R. D. Wolfinger. 1996. *SAS System for Mixed Models*. SAS Institute, Inc., Cary, NC.

Lyytinen, K., D. Robey. 1999. Learning failure in information systems development. *Inform. Systems J.* **9**(2) 85–101.

Mannix, E., M. A. Neale. 2005. What differences make a difference? The promise and reality of diverse teams in organizations. *Psych. Sci. Public Interest* **6**(2) 31–55.

March, J. G. 1991. Exploration and exploitation in organizational learning. *Organ. Sci.* **2**(1) 71–87.

Matusik, S. F., M. B. Heeley. 2005. Absorptive capacity in the software industry: Identifying dimensions that affect knowledge and knowledge creation activities. *J. Management* **31**(4) 549–572.

Meyerson, D., K. E. Weick, R. M. Kramer. 1996. Swift trust and temporary groups. R. M. Kramer, T. R. Tyler, eds. *Trust in Organizations: Frontiers of Theory and Research*. Sage, Thousand Oaks, CA, 166–195.

Pisano, G. P., R. M. J. Bohmer, A. C. Edmondson. 2001. Organizational differences in rates of learning: Evidence from the adoption of minimally invasive cardiac surgery. *Management Sci.* **47**(6) 752–768.

Rajlich, V. 2006. Changing the paradigm of software engineering. *Comm. ACM* **49**(8) 67–70.

Ramanujan, S., R. W. Scamell, J. R. Shah. 2000. An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort. *J. Systems Software* **54**(2) 137–157.

Reagans, R., L. Argote, D. Brooks. 2005. Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together. *Management Sci.* **51**(6) 869–881.

Robillard, P. N. 1999. The role of knowledge in software development. *Comm. ACM* **42**(1) 87–92.

Rulke, D., J. Galaskiewicz. 2000. Distribution of knowledge, group network structure, and group performance. *Management Sci.* **46**(5) 612–625.

Sacks, M. 1994. *On-the-Job Learning in the Software Industry*. Quorum Books, Westport, CT.

SAS Institute. 2000. SAS OnlineDoc, Version 8. SAS Institute, Inc., Cary, NC, http://v8doc.sas.com/sashtml/.

Schilling, M. A., P. Vidal, R. E. Ployhart, A. Marangoni. 2003. Learning by doing something else: Variation, relatedness, and the learning curve. *Management Sci.* **49**(1) 39–56.

Schmidt, R. A. 1975. A schema theory of discrete motor skill learning. *Psych. Rev.* **82**(4) 225–260.

Simonton, D. 1999. Creativity as blind variation and selective retention: Is the creative process Darwinian? *Psych. Inquiry* **10**(4) 309–328.

Singer, J. D. 2002. Fitting individual growth models using SAS PROC MIXED. D. S. Moskowitz, S. L. Hershberger, eds. *Modeling Intraindividual Variability with Repeated Measures Data: Methods and Applications*. L. Erlbaum Associates, Mahwah, NJ, 135–170.

Thornton, R. A., P. Thompson. 2001. Learning from experience and learning from others: An exploration of learning and spillovers in wartime shipbuilding. *Amer. Econom. Rev.* **91**(5) 1350–1368.

Tyre, M. J., E. von Hippel. 1997. The situated nature of adaptive learning in organizations. *Organ. Sci.* **8**(1) 71–83.

Wastell, D. 1999. Learning dysfunctions in information systems development: Overcoming the social defences with transitional objects. *MIS Quart.* **23**(4) 581–600.

Weber, R. A., C. F. Camerer. 2003. Cultural conflict and merger failure: An experimental approach. *Management Sci.* **49**(4) 400–415.

Wegner, D. M. 1986. Transactive memory: A contemporary analysis of the group mind. B. Mullen, G. R. Goethals, eds. *Theories of Group Behavior*. Springer-Verlag, New York, 185–205.

Yelle, L. E. 1979. The learning curve: Historical review and comprehensive survey. *Decision Sci.* **10**(2) 302–328.

Zander, U., B. Kogut. 1995. Knowledge and the speed of the transfer and imitation of organizational capabilities—An empirical-test. *Organ. Sci.* **6**(1) 76–92.

MANAGEMENT SCIENCE
DOI 10.1287/mnsc.1060.0687ec
pp. ec1–ec10

Electronic Companion—"Learning from Experience in Software Development: A Multilevel Analysis" by Wai Fong Boh, Sandra A. Slaughter, and J. Alberto Espinosa, *Management Science*, DOI 10.1287/mnsc.1060.0687.

Online Appendices

Appendix A. Operationalization of Variables for All Three Levels of Analysis

	Operationalization for level of analysis:		
Variable	Individual developer MRs	Group MRs	System releases
Effort per MR/System Release	Total number of hours to complete the individual MR.	Total number of hours to complete the group MR.	Total number of hours to complete the system release. i.e. sum of the imputed effort of all MRs included in the system release
Same-System Experience	Number of MRs completed in the current system by the individual prior to starting work on the current MR	Average number of MRs completed in the current system by individuals working on the group MR prior to starting work on the current MR	Average number of MRs completed in the current system by individuals in the organizational unit working on the system release prior to starting work on the current system release
Related-System Experience	Number of MRs completed in other related systems (relative to the system of the current MR) by the individual prior to starting work on the current MR	Average number of MRs completed in other related systems (relative to the system of the current MR) by individuals working on the group MR prior to starting work on the current MR	Average number of MRs completed in other related systems (relative to the system of the current MR) by developers in the organizational unit working on the current system release prior to starting work on the system release
Unrelated-System Experience	Number of MRs completed in other unrelated systems (relative to the system of the current MR) by the individual prior to starting work on the current MR	Average number of MRs completed in other unrelated systems (relative to the system of the current MR) by individuals working on the group MR prior to starting work on the current MR	Average number of MRs completed in other unrelated systems (relative to the system of the current MR) by developers in the organizational unit working on the current system release prior to starting work on the system release

Appendix A (Continued)

	Operationalization for level of analysis:		
Variable	Individual developer MRs	Group MRs	System releases
Experience Working with Other Developers	Total number of distinct developers that an individual has worked with prior to starting work on the current MR	N.A.	N.A.
Average Shared Experience	N.A.	Average number of MRs completed by both members of each dyad included in the group MR prior to starting work on the current group MR, averaged across all dyads in the group	Average number of system releases completed by both members of each dyad in the organizational unit included in the system release prior to starting work on the current system release, averaged across all dyads working on the system release
Time	The number of months from the start of the development of the product that the individual MR was started.	The number of months from the start of the development of the product that the group MR was started.	The number of months from the start of the development of the product that the system release was started.
Size	Lines of codes added, deleted, and changed for the individual MR	Lines of codes added, deleted, and changed for the group MR	Lines of codes added, deleted, and changed for all MRs in the system release
Complexity	The number of files that have to be changed to complete the individual MR	The number of files that have to be changed to complete the group MR	Average file complexity of the MRs in the system release
Priority	Importance of the individual MR as determined by the change committee	Importance of the group MR as determined by the change committee	Average priority of the MRs in the system release as determined by the change committee
Number of Developers	N.A.	Total number of developers working on the group MR	Average number of developers per MR in the organizational unit working on the system release
New Development	Indicates whether the individual MR is a new development, or maintenance work	Indicates whether the group MR is a new development, or maintenance work	Percentage of MRs constituting: (1) new development individual MRs (2) new development group MRs (3) maintenance individual MRs (4) maintenance group MRs in the system release
Group work	N.A.	N.A.	

Appendix B. Effort Computations for Developer Effort

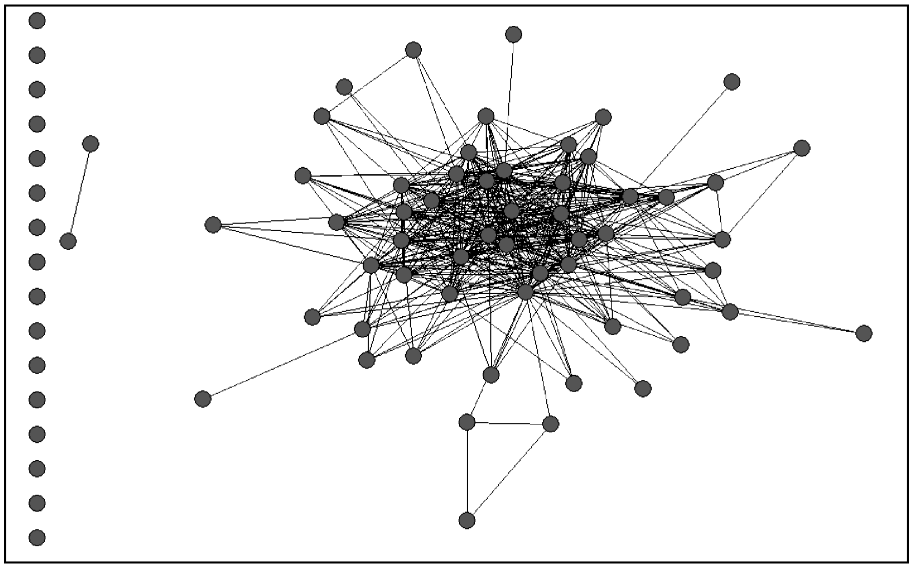
This appendix explains how we impute effort (number of labor hours) expended for each MR, based on the effort-computation algorithm developed in Graves and Mockus (1998) and Atkins et al. (2001), which estimates effort for each MR from information available in the version control and change management systems. Although we have the date and time at which each MR started and ended, this elapsed time is not equivalent to effort, as developers can be working on several MRs at the same time. Nevertheless, by keeping track of the number of MRs that each individual developer works on in each month, we can generate a good estimate of the effort expended per MR. In essence, the algorithm divides each developer's monthly effort equally among the open MRs that the developer is working on as a first estimation. This initial estimation of effort for each MR is then used as the dependent variable measure in analyzing Equation (1) and Equation (2). The regression analyses then provide predicted values of the effort per MR based on the estimated coefficients. This is then used to refine the estimation of effort expended for each MR, and the regression analysis and re-estimation steps are repeated until there is no improvement in the log-likelihood estimates in the analyses. For individual MRs, our analyses converged in the fourth iteration; for group MRs, our analyses converged in the third iteration to produce the effort estimates yielding the optimal log-likelihood results.

To cater to our analysis plan of dividing the data set into individual MRs and group MRs, we made some minor modifications to the Atkins et al. (2001) and Graves and Mockus (1998) algorithms. We illustrate the algorithm using an example for a single developer. Developer A worked on MR A in January, and worked on MR A and MR B in February. We assume that each developer spends 1 unit of time per month, an assumption that has been shown to provide similar results as using monthly time sheet data (Atkins et al. 2001).

Step 1. We divide up the monthly effort equally across all changes open in that month to obtain estimates of total MR efforts for each developer. Developer A spends one unit of effort on MR A in January, 0.5 units of effort on MR A in February, and 0.5 units of effort in February on MR B. The initial estimation of effort is thus 1.5 units of effort for MR A, and 0.5 units of effort for MR B. For group MRs, the initial estimation of effort for all the individuals working on the group MR are added together to form the initial estimation of effort for the group MR.

Step 2. We fit our regression models (Equations (1) and (2)) using the initial estimation of effort as the dependent variable measures for all developers. For individual MRs, the predicted values become the new estimates of MR effort. For group MRs, the predicted values of effort for group MRs

Figure EC.1 Network Diagram Showing How Systems Are Related to One Another



Note. Unconnected dots show systems that are not related to any other systems.

Table EC.1 Descriptive Statistics for Experience Variables of Individuals

Variable	Mean	Std. dev.	Minimum	Maximum
Same-System Experience	34.916	60.411	0.000	1,195.280
Related-Systems Experience	31.183	75.048	0.000	2,443.600
Unrelated-Systems Experience	4.119	15.566	0.000	258.462
Total cumulative experience	70.219	125.155	0.000	2,937.000

Note. Number of MRs = 518,971.

are divided among individuals involved in the group MR based on each individual's initial effort contribution to the group MR (i.e., the proportion of effort, using estimates from Step 1, expended by each individual).[EC1] This is the only step that differs from the Graves and Mockus (1998) algorithm, as the latter did not differentiate between group MRs and individual MRs.

Step 3. Next, we rescale the effort per MR per month for each developer so that it is equal to the predicted values of effort expended for each MR. If the predicted effort for MR A is 2.0, we then impute that developer A spent $(1/1.5 \times 2)$ 1.33 units of effort in January, and 0.67 units of effort in February on MR A. If the predicted value for MR B is 0.5, we impute that developer A spent 0.5 units of effort in February on MR B.

Step 4. We rescale the effort per month per MR so that the total effort expended per month is equal to the observed monthly efforts for each developer. In our example for developer A, effort for MR A is rescaled to 1.0 in January, and $(0.67/(0.67+0.5) \times 1.0)$ 0.57 in February, whereas effort for MR B is rescaled to 0.43 in February (so that effort in February still sums to 1.0 $[0.57+0.43]$).

Iterate. The above four steps are repeated for each iteration, until convergence, which occurs when there is no improvement in the -2 log likelihood in the model fitting step. For our analysis, the -2 log likelihood measure converged at the 4th iteration for individual MRs, and it converged at the 3rd iteration for group MRs. Hence, the imputed effort measures from the 4th iteration were used in the analyses for individual MRs, and the imputed effort measures from the 3rd iteration were used in the analyses for group MRs.

Appendix C. Definition of Related and Unrelated Systems

We examined the extent to which two systems are functionally related by calculating the extent to which both systems have MRs generated by the same initial request. We observed a high extent of variation in the extent to which any two systems are related, where the number of initial requests affecting any two systems can range from zero to 7,227. The mean number of initial requests affecting any two systems is 81. Using this as the cut-off value, we define two systems to be functionally related if they have at least 81 initial requests affecting both systems. Based on this definition, we observed that 19.59% of all possible pairs of systems are considered to be related. Figure EC.1 is a network diagram that shows how the systems are related to one another.

Using this definition of related and unrelated systems, we calculated the mean experience for each individual in terms of the same-system, related-system, unrelated-system experience, and overall MR experience. Table EC.1 provides the descriptive statistics for the different types of experience for all the individuals in the data set.

We conducted sensitivity tests to examine if our results would change if we use different cut-off values to define related vs. unrelated systems. The first test used the mean number of initial requests affecting any two systems plus one standard deviation (334) as the cut-off value, and the second test used the modal number of initial requests affecting any two systems (14) as the cut-off value.[EC2] Our results remain robust across all the sensitivity tests.

[EC1] The estimated effort expended by each individual in a group MR is the predicted amount of effort for the group MR, multiplied by the proportion of effort that each individual puts in for the group based on the estimates in Step 1.

[EC2] We did not use the mean number of initial requests affecting any two systems minus one standard deviation as this cut-off was a negative number.

Appendix D. Correlation Tables for Analyses at Individual, Group, and Organizational-Unit Levels of Analysis

Table EC.2 Correlation Matrix for Individual Developer MRs

Variable	Mean (s.d.)	1	2	3	4	5	6	7	8	9
1. *Ln Effort Per Individual MR*	−1.99 (1.05)									
2. *Ln Size*	3.11 (2.14)	0.16								
3. *Ln File Complexity*	0.42 (0.71)	0.10	0.46							
4. *New Development*	0.41 (0.49)	−0.24	0.08	0.10						
5. *Priority*	2.17 (1.04)	0.21	−0.08	−0.12	−0.49					
6. *Start Month*	153.26 (50.56)	−0.13	−0.07	−0.09	0.04	0.07				
7. *Ln Individual Same-System Experience*	3.28 (3.35)	−0.27	−0.01	−0.02	−0.06	0.06	0.10			
8. *Ln Individual Related-Systems Experience*	2.02 (5.45)	−0.23	−0.07	−0.08	0.07	0.03	0.22	0.14		
9. *Ln Individual Unrelated-Systems Experience*	−6.87 (6.91)	−0.09	−0.01	0.04	0.01	0.03	0.14	0.02	0.21	
10. *Ln Individual Experience Working with Other dev.*	−3.28 (6.33)	−0.24	−0.01	0.04	0.09	−0.08	−0.14	0.08	0.14	0.11

Notes. All correlations with absolute value > 0.003 are significant at $p < 0.001$; number of observations (number of MRs) = 518,971.

Table EC.3 Correlation Matrix for Group MRs

Variables	Mean (s.d.)	1	2	3	4	5	6	7	8	9	10
1. *Ln Effort Per Group MR*	0.19 (1.16)										
2. *Ln Size*	7.19 (6.09)	0.53									
3. *Ln File Complexity*	1.92 (1.25)	0.70	0.37								
4. *Ln Number of Developers in Group MR*	0.84 (0.28)	0.50	0.22	0.44							
5. *New Development*	0.35 (0.48)	0.16	0.11	0.22	0.16						
6. *Priority*	1.28 (1.27)	−0.11	−0.02	−0.10	0.00	0.15					
7. *Start Month*	139.61 (52.36)	−0.17	0.00	−0.04	−0.07	0.02	0.04				
8. *Ln Average Group Same-System Experience*	3.45 (2.76)	−0.17	−0.14	0.02	0.04	−0.06	0.04	0.16			
9. *Ln Average Group Related-Systems Experience*	3.20 (3.85)	−0.23	−0.03	−0.08	0.01	0.16	0.20	0.25	0.16		
10. *Ln Average Group Unrelated-Systems Experience*	−3.92 (6.98)	−0.14	−0.03	0.01	0.07	0.14	0.14	0.21	0.04	0.28	
11. *Ln Average Shared Group Experience*	−5.04 (6.37)	−0.20	−0.24	0.06	0.17	−0.02	0.01	0.07	0.28	0.07	0.10

Notes. All correlations with absolute value > 0.018 are significant at $p < 0.001$; number of observations (number of group MRs) = 30,225.

Table EC.4 Correlation Matrix for System Releases

Variables	Mean (s.d.)	1	2	3	4	5	6	7	8	9	10	11	12
1. *Ln Effort per System Release*	2.78 (2.15)												
2. *Ln Size*	8.95 (3.29)	0.84											
3. *Ln Average File Complexity*	0.98 (0.59)	0.12	0.30										
4. *Ln Percent New group MRs*	−2.43 (3.30)	0.55	0.55	0.17									
5. *Ln Percent Maintenance Individual MRs*	−1.72 (2.86)	0.43	0.44	0.02	0.15								
6. *Ln Percent Maintenance Group MRs*	−4.82 (4.39)	0.58	0.54	0.11	0.31	0.25							
7. *Ln Number of Developers per MR*	0.37 (0.31)	0.21	0.28	0.25	0.49	−0.14	0.35						
8. *Ln Average Priority*	0.52 (1.22)	0.20	0.19	0.02	0.17	0.11	0.09	−0.02					
9. *Start Month*	133.71 (53.92)	−0.13	−0.15	−0.13	−0.15	−0.09	−0.08	−0.11	−0.03				
10. *Ln Average Unit Same-System Experience*	2.77 (2.03)	0.29	0.29	−0.05	0.25	0.19	0.24	0.16	0.09	0.20			
11. *Ln Average Unit Related-Systems Experience*	2.84 (5.29)	0.26	0.18	−0.11	0.23	0.10	0.25	0.08	0.00	0.05	0.30		
12. *Ln Average Unit Unrelated-Systems Experience*	2.92 (2.79)	−0.08	−0.07	−0.02	−0.04	0.15	−0.01	−0.09	0.03	0.27	−0.02	−0.08	
13. *Ln Average Shared-System Release Experience*	6.17 (4.72)	0.47	0.38	−0.03	0.40	0.19	0.37	0.22	0.06	0.09	0.29	0.29	0.23

Notes. All correlations with absolute value > 0.06 are significant at $p < 0.001$; number of observations (number of system releases) = 2,282.

Appendix E. Sensitivity Analysis to Check for Potential Multicollinearity Problems

We conducted sensitivity analysis at the individual, group, and organizational unit levels of analyses to check for potential multicollinearity problems. For each level of analysis, we conducted the following analyses:

(1) We first reviewed the correlation tables (Tables EC.2, EC.3, and EC.4) to ensure that the correlations between the independent variables were not excessively high.

(2) We re-ran each analysis using ordinary least squares (OLS) so as to obtain the VIF and Condition Indices (CIs). The variance inflation factor (VIF) and CI for each level of analysis are provided in Tables EC.5, EC.6, and EC.7.

(3) For each level of analysis, we test and demonstrate the robustness of our model by adding the control variables one by one into our main model to show that the results of our analyses are stable and are not affected by dropping control variables.

We discuss the results for each level of analysis below.

1. Dependent Variable: Productivity for Individual MRs

1.1. Correlations Amongst Independent Variables. Table EC.2 shows that none of the correlations amongst the independent variables are higher than 0.5. Only two correlations are higher than 0.4: (1) Individual MRs that are new feature development are likely to have lower priorities ($r = -0.493$); and (2) MRs that are bigger in size are likely to be more complex ($r = 0.462$).

1.2. VIFs and Condition Index. The last column in Table EC.5 shows the VIFs for the independent variables. The highest VIF in this analysis is only 1.357, and the CI of 13.674 is considerably low. These figures suggest that multicollinearity is not a problem for the individual level of analysis.

Table EC.5 Robustness Analysis for Individual Level Analysis

Variables	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	VIF
Intercept	−1.589*** (0.013)	−1.775*** (0.013)	−1.775*** (0.013)	−1.640*** (0.013)	−1.879*** (0.013)	−1.768*** (0.019)	
Ln Size (β_1)		0.055*** (0.000)	0.048*** (0.000)	0.050*** (0.000)	0.050*** (0.000)	0.050*** (0.000)	1.278
Ln File Complexity (β_2)			0.048*** (0.001)	0.064*** (0.001)	0.068*** (0.001)	0.068*** (0.001)	1.300
New Development (β_3)				−0.366*** (0.002)	−0.273*** (0.002)	−0.273*** (0.002)	1.357
Priority (β_4)					0.092*** (0.001)	0.092*** (0.001)	1.351
Start Month (β_5)						−0.001*** (0.000)	1.122
Ln Individual Same-System Experience (β_6)	−0.033*** (0.001)	−0.032*** (0.001)	−0.032*** (0.001)	−0.033*** (0.001)	−0.034*** (0.001)	−0.033*** (0.001)	1.038
Ln Individual Related-Systems Experience (β_7)	−0.030*** (0.001)	−0.029*** (0.001)	−0.029*** (0.001)	−0.027*** (0.001)	−0.027*** (0.001)	−0.026*** (0.001)	1.144
Ln Individual Unrelated-Systems Experience (β_8)	−0.010*** (0.001)	−0.011*** (0.001)	−0.011*** (0.001)	−0.010*** (0.001)	−0.011*** (0.001)	−0.009*** (0.001)	1.072
Ln Individual Experience Working with other dev (β_9)	−0.017*** (0.001)	−0.018*** (0.001)	−0.018*** (0.001)	−0.015*** (0.001)	−0.015*** (0.001)	−0.013*** (0.001)	1.081
Deviance (−2 log likelihood)	771,015.8	738,189.8	736,008.1	696,024.1	686,870.8	685,662	CI:13.674
Within-individual variance	0.568	0.5612	0.5615	0.5241	0.5164	0.5165	
Deviance difference (Δdev)	755,713	32,826	2,182	39,984	9,153	1,209	
Proportion of within-individual variance explained (%)	48.81	49.42	49.40	52.77	53.46	53.45	

Notes. (1) Number of observations (number of MRs) = 518,971; Number of subjects for random effects (number of individuals) = 5,123; number of systems included in fixed-effects = 89. (2) Within-subject correction for AR(1) error structure and individual random effects are included for all models.

Table EC.6 Robustness Analysis for Group Level Analysis

Variables	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	VIF
Intercept	0.302***	−0.303***	−1.211***	−1.954***	−1.974***	−1.960***	−1.761***	
	(0.067)	(0.058)	(0.042)	(0.038)	(0.038)	(0.038)	(0.039)	
Ln Size (β_1)		0.086***	0.040***	0.034***	0.034***	0.034***	0.034***	1.301
		(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
Ln File Complexity (β_2)			0.556***	0.457***	0.448***	0.447***	0.446***	1.471
			(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	
Ln Number of Developers In Group MR (β_3)				1.148***	1.137***	1.139***	1.119***	1.308
				(0.013)	(0.013)	(0.013)	(0.013)	
New development (β4)					0.105***	0.107***	0.106***	1.135
					(0.007)	(0.007)	(0.007)	
Priority (β_5)						−0.010***	−0.011***	1.082
						(0.003)	(0.003)	
Start Month (β_6)							−0.001***	1.125
							(0.000)	
Ln Avg. Group Same-System Experience (β_7)	−0.032***	−0.019***	−0.029***	−0.028***	−0.027***	−0.027***	−0.024***	1.146
	(0.002)	(0.002)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
Ln Avg. Group Related-Systems experience (β_8)	−0.052***	−0.054***	−0.037***	−0.036***	−0.038***	−0.038***	−0.034***	1.216
	(0.002)	(0.002)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
Ln Avg. Group Unrelated-Systems Experience (β_9)	−0.003**	−0.006***	−0.012***	−0.016***	−0.017***	−0.017***	−0.016***	1.144
	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
Ln Avg. Shared-Group Experience (β_{10})	−0.016***	−0.003**	−0.019***	−0.028***	−0.027***	−0.027***	−0.027***	1.210
	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
Deviance (−2 log likelihood)	86,044.1	78,021.9	57,094.2	50,276.2	50,070.7	50,058.2	49,802.6	CI:15.439
Within-system variance	1.0133	0.7765	0.3889	0.3098	0.3077	0.3075	0.3047	
Deviance difference (Δdev)	8,803	8,022	20,928	6,818	206	13	256	
Proportion of within-system variance explained (%)	24.95	42.49	71.19	77.05	77.21	77.22	77.43	

Notes. (1) Number of observations (no. of group MRs) = 30,225; number of systems included in fixed effects = 79, df = 30,000. (2) Within-subject correction for AR(1) error structure included for all models.

1.3. Test of Model Robustness. Table EC.5 shows the results of our robustness analysis for individual MRs, where control variables are added one at a time to our model. The results of the key variables—the experience variables—are stable and are not affected by dropping control variables.

2. Dependent Variable: Productivity for Group MRs

2.1. Correlations Amongst Independent Variables. Table EC.3 shows that none of the correlations amongst the independent variables are higher than 0.5. Only one correlation is higher than 0.4: group MRs that have are higher in complexity are likely to involve more developers ($r = 0.44$).

2.2. VIFs and Condition Index. The last column in Table EC.6 shows the VIFs for the independent variables. The highest VIF in this analysis is only 1.471, and the CI of 15.439 is also considerably low. These figures suggest that multicollinearity is not a problem for the group level of analysis.

2.3. Test of Model Robustness. Table EC.6 shows the results of our robustness analysis for group MRs, where control variables are added one at a time to our model. The results of the key variables—the experience variables—are stable and are not affected by dropping control variables. Additionally, there are no slip flips between models for any of the variables in the models.

3. Dependent Variable: Productivity for System Release

3.1. Correlations Amongst Independent Variables. Table EC.4 in Appendix D shows that none of the correlations amongst the independent variables are higher than 0.55. Only three correlations are higher than 0.4: (1) system releases that are larger in size tend to have a greater percentage of maintenance individual MRs ($r = 0.44$); and (2) system releases that are larger in size tend to have a greater percentage of group MRs, both maintenance group MRs ($r = 0.54$) and new development group MRs ($r = 0.55$).

Table EC.7 Robustness Analysis for Organizational-Unit Level Analysis

Variables	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	VIF
Intercept	2.589*** (0.471)	−1.104*** (0.266)	−0.946*** (0.259)	0.695 (0.260)	0.975 (0.257)	0.977*** (0.257)	1.016*** (0.262)	
Ln size (γ_1)		0.480*** (0.007)	0.505*** (0.008)	0.407*** (0.010)	0.403*** (0.009)	0.401*** (0.009)	0.400*** (0.009)	2.471
Ln Average File Complexity (γ_2)			−0.395*** (0.040)	−0.365*** (0.038)	−0.299*** (0.038)	−0.296*** (0.038)	−0.297*** (0.038)	1.232
Ln Percent New Group MRs (γ_3)				0.045*** (0.006)	0.066*** (0.006)	0.065*** (0.006)	0.064*** (0.006)	1.957
Ln Percent Maintenance Individual MRs (γ_4)				0.069*** (0.008)	0.051*** (0.008)	0.052*** (0.008)	0.051*** (0.008)	1.470
Ln Percent Maintenance Group MRs (γ_5)				0.072*** (0.006)	0.085*** (0.006)	0.085*** (0.006)	0.085*** (0.006)	1.653
Ln Avg. No. of Developers Per MR (γ_6)					−0.727*** (0.086)	−0.709*** (0.087)	−0.712*** (0.087)	1.660
Ln Avg. Priority (γ_7)						0.033* (0.016)	0.033* (0.016)	1.073
Start Month (γ_8)							−0.000 (0.000)	1.238
Ln Avg. Unit Same-System Experience (γ_9)	0.112*** (0.022)	−0.012 (0.013)	−0.006 (0.013)	−0.015 (0.012)	−0.016 (0.012)	−0.017 (0.012)	−0.014 (0.012)	1.319
Ln Avg. Unit Related-Systems Experience (γ_{10})	−0.148*** (0.025)	−0.060*** (0.015)	−0.054*** (0.015)	−0.055*** (0.014)	−0.055*** (0.014)	−0.055*** (0.014)	−0.053*** (0.014)	1.242
Ln Avg. Unit Unrelated-Systems Experience ($\gamma 11$)	−0.007 (0.017)	0.009 (0.010)	0.006 (0.010)	−0.010 (0.009)	−0.011 (0.009)	−0.011 (0.009)	−0.010 (0.009)	1.267
Ln Avg. Shared-System Release Experience ($\gamma 12$)	0.188*** (0.009)	0.076*** (0.006)	0.073*** (0.005)	0.056*** (0.005)	0.056*** (0.005)	0.056*** (0.005)	0.056*** (0.005)	1.545
Deviance (−2 log likelihood)	8,779.9	6,449.5	6,354.9	6,092.4	6,022.3	6,018.2	6,017.6	CI: 22.367
Within-system variance	2.7583	0.9889	0.9492	0.8479	0.8229	0.8213	0.8209	
Deviance difference (Δdev)	1,187	2,330	95	263	70	4	1	
Proportion of within-system variance explained (%)	40.16	78.55	79.41	81.60	82.15	82.18	82.19	

Notes. (1) Number of observations (number of system releases) = 2,282; number of systems included in fixed effects = 68, df = 2,202. (2) Within-subject correction for AR(1) repeated measures error structure included for all models.

3.2. VIFs and Condition Index. The last column in Table EC.7 below shows the VIFs for the independent variables. The highest VIF in this analysis is only 2.471, and the CI of 22.367 is also reasonably low. These figures suggest that multicollinearity is not a problem for the organizational unit level of analysis.

3.3. Test of Model Robustness. Table EC.7 shows the results of our robustness analysis for system releases, where control variables are added one at a time to our model. The results for *average unit related-systems experience*, and for *average shared system experience* are significant and consistent across all the models. The results for *average unit unrelated-systems experience* are consistently insignificant across all the models. The coefficient for *average unit same-system experience* became insignificant after the system release size is controlled for (Model 2), and remained insignificant for the rest of the models. This is likely because system releases with higher average unit same-system experience also tend to be larger ($r = 0.29$), and system releases that are larger tend to require more effort to develop ($r = 0.84$). Hence, the average individual same-system experience showed a positive and significant coefficient in Model 1, and the coefficient became insignificant once the release size is controlled for. Note also that when the percentage variables distinguishing between new development and maintenance for individual and group MRs are added to the analysis in Model 4, the intercept changes to reflect its new role in the analysis as the base case for these percentage variables (specifically, it reflects the value of the coefficient for *New Development Individual MRs*).

Due to the slightly higher level of CI at the organizational unit level of analysis, we conducted further robustness checks to ensure that the results were not driven by correlations amongst the

Table EC.8 Additional Robustness Analysis for Organizational-Unit Level Analysis

Variables	Model 1	Model 2	Model 3	Model 4	Model 5
Intercept	1.016***	1.087***	1.193***	1.073***	0.815***
	(0.262)	(0.265)	(0.269)	(0.266)	(0.260)
Ln size (γ_1)	0.400***	0.415***	0.413***	0.415***	0.405***
	(0.009)	(0.010)	(0.010)	(0.010)	(0.009)
Ln Average File Complexity (γ_2)	−0.297***	−0.335***	−0.332***	−0.334***	−0.301***
	(0.038)	(0.039)	(0.039)	(0.039)	(0.038)
Ln Percent New Group MRs (γ_3)	0.064***	0.077***	0.078***	0.076***	0.063***
	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)
Ln Percent Maintenance Individual MRs (γ_4)	0.051***	0.048***	0.051***	0.047***	0.043***
	(0.008)	(0.008)	(0.008)	(0.008)	(0.008)
Ln Percent Maintenance Group MRs (γ_5)	0.085***	0.096***	0.096***	0.095***	0.084***
	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)
Ln Avg. No. of Developers Per MR (γ_6)	−0.712***	−0.706***	−0.707***	−0.706***	−0.709***
	(0.087)	(0.089)	(0.089)	(0.089)	(0.087)
Ln avg. priority (γ_7)	0.033*	0.030+	0.029+	0.029+	0.032+
	(0.016)	(0.017)	(0.017)	(0.017)	(0.016)
Start Month (γ_8)	−0.000	−0.000	−0.000	−0.000	−0.001**
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
Ln Avg. Unit Same-System Experience (γ_9)	−0.014	−0.003			
	(0.012)	(0.013)			
Ln Avg. Unit Related-Systems Experience (γ_{10})	−0.053***		−0.036**		
	(0.014)		(0.013)		
Ln Avg. Unit Unrelated-Systems Experience (γ_{11})	−0.010			−0.004	
	(0.009)			(0.008)	
Ln Avg. Shared-System Release Experience (γ_{12})	0.056***				0.049***
	(0.005)				(0.005)
Deviance (−2 log likelihood)	6,017.6	6,138.4	6,114.1	6,138.2	6,042.1
Within-system variance	0.8209	0.8649	0.8534	0.8648	0.8294
Proportion of within-system variance explained (%)	82.19	81.24	81.48	81.24	82.01

experience variables, by adding in the experience variables one at a time to the base model with only the control variables. Table EC.8 below shows the results of this analysis.

The results in Table EC.8 show that the results for the experience variables remain the same when added into the base model one at a time, indicating that the coefficients of the experience variables were indicative of the true relationship between the experience variables and the dependent variable, after controlling for size and other key characteristics of the system release. The results, therefore, appear to be robust, and not subject to multicollinearity issues.

Appendix F. Additional Sensitivity Analysis for New and Maintenance MR Experience

We conducted additional sensitivity analyses to examine whether the experience in new development MRs differs from experience in maintenance MRs, in terms of productivity benefits. We coded two new variables: *New MRs Experience* and *Maintenance MRs Experience*. *New MRs Experience* refers to the cumulative number of new development MRs that each individual has completed prior to the start of the current MR. *Maintenance MRs Experience* refers to the cumulative number of maintenance MRs that each individual has completed prior to the start of the current MR.

We included these two additional experience variables in the analyses for individual and group MRs. We did not include the two variables at the organizational unit level of analysis due to the significant increase in VIFs and Collinearity Indices caused by the high correlation between these two variables at this level of analysis. Tables EC.9 and EC.10 show the results of the new models at the individual and group levels of analyses, and how they compare with the original results that do not include these two additional experience variables.

At the individual level of analysis, we found that including these two additional experience variables did not change our results. We note that both new and maintenance MRs experience significantly

Table EC.9 Including New Experience Variables for Individual MRs

Variables	Original model	New model
Intercept	-1.768^{***} (0.019)	-1.826^{***} (0.019)
Ln size (β_1)	0.050^{***} (0.000)	0.050^{***} (0.000)
Ln File Complexity (β_2)	0.068^{***} (0.001)	0.068^{***} (0.001)
New Development (β_3)	-0.273^{***} (0.002)	-0.272^{***} (0.002)
Priority (β_4)	0.092^{***} (0.001)	0.092^{***} (0.001)
Start Month (β_5)	-0.001^{***} (0.000)	0.000^{**} (0.000)
Ln Individual Same-System Experience (β_6)	-0.033^{***} (0.001)	-0.032^{***} (0.001)
Ln Individual Related-Systems Experience (β_7)	-0.026^{***} (0.001)	-0.024^{***} (0.001)
Ln Individual Unrelated-Systems Experience (β_8)	-0.009^{***} (0.001)	-0.010^{***} (0.001)
Ln Individual Experience Working with Other dev (β_9)	-0.013^{***} (0.001)	-0.015^{***} (0.001)
Ln Individual New MRs Experience		-0.005^{***} (0.001)
Ln Individual Maintenance MRs Experience		-0.003^{***} (0.001)
Deviance (-2 log likelihood)	685,662	685,395
Within-individual variance	0.5165	0.5162
Deviance difference (Δdev)		267
Proportion of within-individual variance explained (%)	53.45	53.48

Table EC.10 Including New Experience Variables for Group MRs

Variables	Original model	New model
Intercept	-1.761^{***} (0.039)	-1.737^{***} (0.040)
Ln Size (β_1)	0.034^{***} (0.001)	0.034^{***} (0.001)
Ln File Complexity (β_2)	0.446^{***} (0.003)	0.446^{***} (0.003)
Ln Number of Developers in Group MR (β_3)	1.119^{***} (0.013)	1.117^{***} (0.013)
New Development (β_4)	0.106^{***} (0.007)	0.110^{***} (0.007)
Priority (β_5)	-0.011^{***} (0.003)	-0.012^{***} (0.003)
Start Month (β_6)	-0.001^{***} (0.000)	-0.001^{***} (0.000)
Ln Avg. Group Same-System Experience (β_7)	-0.024^{***} (0.001)	-0.022^{***} (0.001)
Ln Avg. Group Related-Systems Experience (β_8)	-0.034^{***} (0.001)	-0.032^{***} (0.001)
Ln Avg. Group Unrelated-Systems Experience (β_9)	-0.016^{***} (0.001)	-0.015^{***} (0.001)
Ln Avg. Shared Group Experience (β_{10})	-0.027^{***} (0.001)	-0.027^{***} (0.001)
Ln Avg. Group New MRs Experience		-0.010^{***} (0.002)
Ln Avg. Group Maintenance MRs Experience		0.001 (0.002)
Deviance (-2 log likelihood)	49,802.6	49,781.3
Within-system variance	0.3047	0.3045
Deviance difference (Δdev)		21
Proportion of within-system variance explained (%)	77.43	77.45

increased the productivity of individual MRs, in addition to the same-system, related- , and unrelated-system variables currently included in the model. At the group level, including these two variables again did not change our results. Only the *New MRs Experience* variable turned out to be significant in decreasing the effort per group MR. The results show that our key results remain unchanged even when we include additional experience variables to differentiate between new and repair MR experience. In addition, the proportion of additional variance explained was only marginal and the size of the coefficients for the two new experience variables was much smaller than the size of the coefficients of the original set of experience variables for both the individual and group levels of analysis. This suggests that the particular type of MR experience (whether new development or maintenance) is not as relevant as the total MR experience in the same, related, or unrelated systems.

Reference

Atkins, D., T. Ball, T. Graves, A. Mockus. 2001. Using version control data to evaluate the impact of software tools: A case study of the version editor. *IEEE Trans. Software Engrg.* **28**(7) 625–637.